Fast Maneuvering Spacecraft with RCS and Single Gimbal CMGs

In this example we have a spacecraft that requires high precision pointing for optical instruments, such as a telescope, or laser beam. The spacecraft Line-of-Sight (LOS) is along the body x-axis and it is pointing at targets. It is pointed at a target for a period of time and then it rotates to point at different target, and then another. The spacecraft is described as agile because under normal operations it is constantly maneuvering between targets and the retargeting must be completed as fast as possible. The on-board Attitude Control System (ACS) is operating in different control modes depending on the circumstances. It uses a combination of RCS jets and Single-Gimbal Control Moment Gyros (SG-CMG). In normal mode of operation, the ACS uses

four Control Moment Gyros which provide high torque required for fast maneuvering between targets. The CMGs use solar energy and they do not require fuel, they also provider smoother operation than RCS. The spacecraft uses the RCS jets for momentum desaturation and also as an ACS backup. It also has a fixed thruster engine for orbital maneuvering. The spacecraft structure is flexible because of the solar arrays and other communication appendages that require finite element modeling and detailed flex analysis. To further complicate the analysis, there is also a propellant tank containing fuel for the main engine and the RCS jets. The propellant sloshing inside the tank during orbital maneuvering affects the vehicle stability and introduces oscillatory disturbances that require analysis because they may degrade the LOS pointing accuracy.

The following analysis focuses mainly in the two main modes of operation, the RCS and the CMG attitude control. We will develop several dynamic models for this flexible spacecraft, design control laws and analyze the ACS stability and performance in various modes of operation. The analysis begins with simple rigid-body models and it gradually becomes more complex as we include structural flexibility and fuel sloshing dynamics. We also gradually increase the complexity of the control laws, starting with a simple phase-plane 3-dof logic and upgrading it to a 6-dof logic that provides simultaneous translation and attitude control. The modified logic is also designed to reduce fuel consumption by pulse-width-modulating the RCS jets. Other control ideas, such as, using blended CMGs and Reaction Wheels (RW), and combined RCS with RW configurations are also evaluated.

In section 1 we use the Flixan program to prepare various flexible spacecraft dynamic models that will be used in later chapters. Two sets of models are created using the “Flexible Spacecraft FEM” program (FEM), and the “Flight Vehicle Modeling” program (FVP), and they are saved in two separate folders. The modeling section can be skipped if the user is already familiar with vehicle modeling in Flixan and may jump to the more interesting analysis sections. In section 2 we design and analyze the RCS system. We begin with a simple 3-dof phase-plane, combined with a dot-product jet-selection logic, and gradually upgrade it into a more advanced logic that minimizes fuel usage. The method is augmented to 6-dof by including also translational control. In section 2.6 we develop a non-linear slosh model for a partially filled fuel tank. This model is used for zero g or low g environment. It is combined with the spacecraft model and used to perform stability analysis. In section 3 we develop the Max Energy, non-linear control law for a 4 SG-CMG array with a singularity avoidance algorithm and implement it in various simulation models. Finally, in chapter 4 we present simulation models that demonstrate multi-mode operations.

1.0 Preparation of the Structure Flexibility Models

Structural flexibility is an important factor to consider when analyzing stability and performance of this spacecraft. It has solar arrays, antennas, and other appendages mounted on its relatively solid bus structure that create low damped flex resonances which induce disturbances on the spacecraft when they get excited by the spacecraft motion as it maneuvers around. This causes degradation in LOS pointing, and jitter in optical imaging. Flexibility may also cause structural instability if not filtered properly. In this section we are describing how to use the Flixan program to generate linear models of the spacecraft that include structural flexibility and fuel sloshing. If you are already familiar with the Flixan modeling process you may bypass this section and go to the more interesting stuff in Section 2.

The first 46 structural modes of this spacecraft are saved in file "*FlexSc_Rcs.Mod*". They were obtained from a finite element modeling (FEM) program. The title of the modal data is "*Flexible Spacecraft with Solar Array, RCS and CMGs*". There is a block of data for each mode (resonance) and there are 46 blocks of data. Each block contains the mode shapes and slopes of a particular resonance at 19 vehicle locations (nodes). The first six modes are rigid-body modes at zero frequency and they define the rigid-body motion of the spacecraft. We typically throw away the first 6 rigid-body modes because we have our own rigid-body models which are more efficient. The real structural modes start with mode number 7 which is at 0.5 Hz. Another important file for flex model preparation is the nodes map, file "*FlexSc_Rcs.Nod*", which describes the identity of the 19 structural locations (nodes) included in the modal data file, in the same order as they are listed there. The map file is used in menus by the model preparation programs.

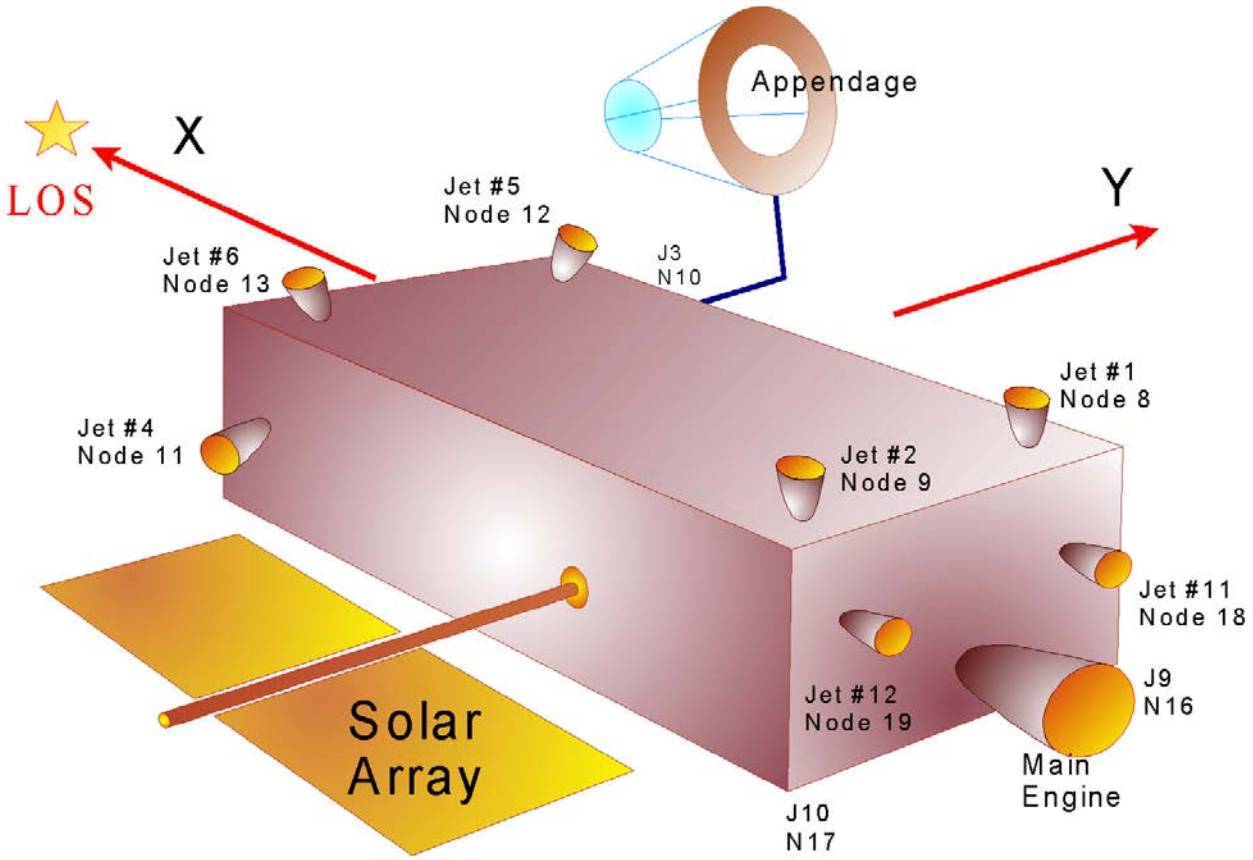


Figure 1 Location of the RCS Jets and the Main Engine

1.1 Flex Models Created using the Flex Spacecraft Modeling Program

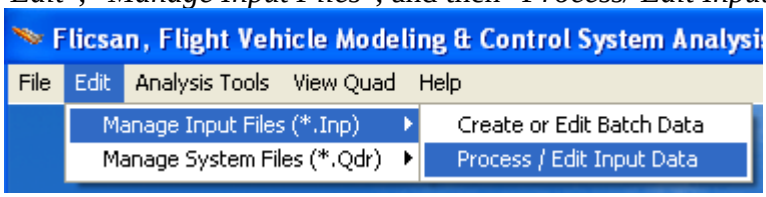
In this section we will describe the preparation of some of the flex spacecraft state-space systems that will be used in the subsequent analysis sections. The systems are created from an already existing finite elements model using the "Flex Spacecraft FE Modeling Program" from the Flixan package. The files for this model preparation are in folder "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\(a) Flex Models from Spacecraft FEM*". This folder in addition to the modal data and node files it contains also the spacecraft input data file "*FlexSc_FEM.Inp*", which contains the data for creating three spacecraft models using the Flex Spacecraft FE Modeling Program. The three spacecraft systems are:

1. A rigid-body model created from the first 6 modes of the FEM, which are rigid-body modes. Its title is "*Rigid-Body Spacecraft with RCS and CMG*".
2. A flexible spacecraft model that uses all 46 FEM modes, 6 rigid-body and 40 flex. Its title is "*RB+Flex Spacecraft with RCS and CMG*", and
3. A structural model only, which uses only the 40 flex modes, excluding the first 6 rigid modes. Its title is "*Flex Only Spacecraft with RCS and CMG*".

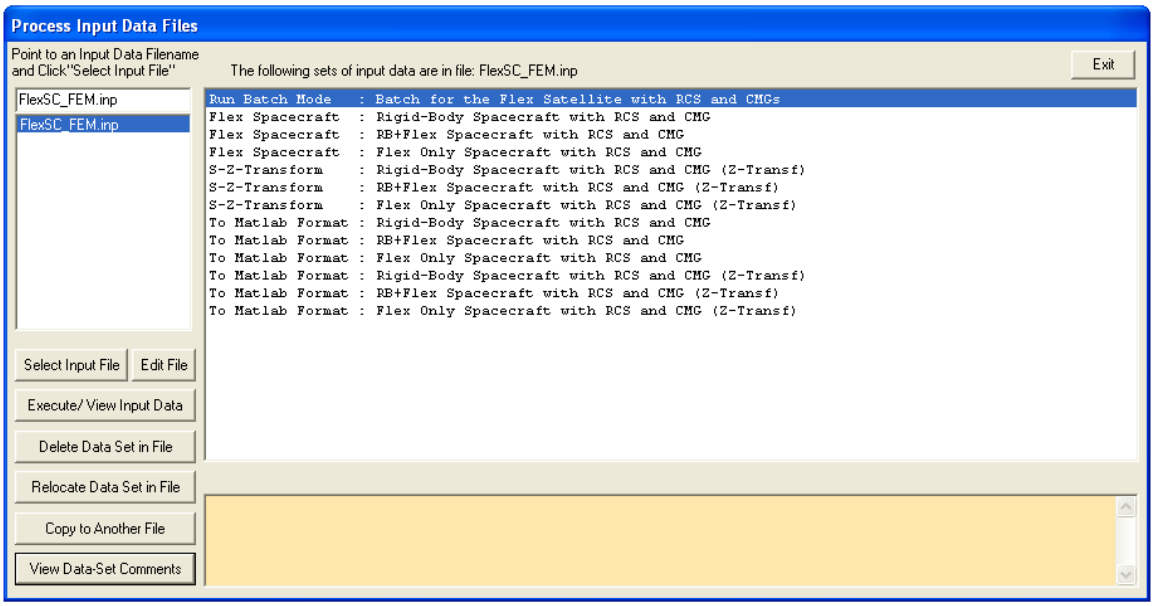
On the top of the input data file there is a batch data-set "*Batch for the Flex Satellite with RCS and CMGs*" which is a short script of commands that automates the generation of the spacecraft systems. The above batch creates the three spacecraft state-space systems, discretizes them using 5 msec sample rate, and converts them in Matlab format for the analysis that follows. Although the data for creating the spacecraft models is already in file "*FlexSc_FEM.Inp*" and all you have to do is run the batch file to create the models, in the following two sections we will demonstrate how to create the input data-sets using both approaches: from the batch and also from scratch.

Creating the Systems in Batch Mode

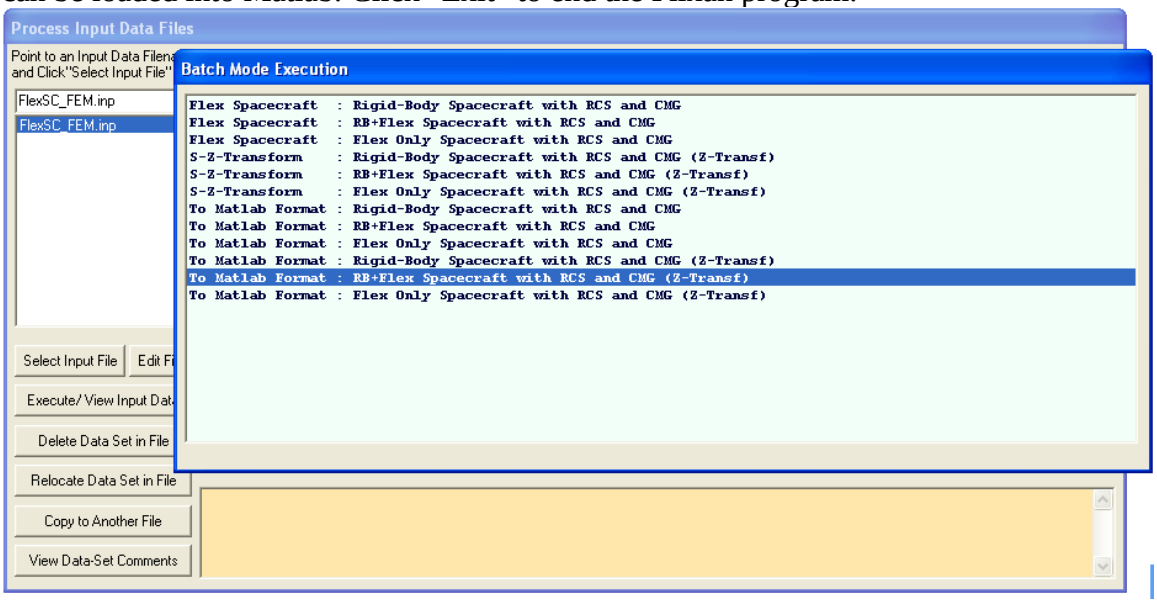
To run the batch set, first start the Flixan program, go to folder "*… \Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\ (a) Flex Models from Spacecraft FEM*". Go to "*Edit*", "*Manage Input Files*", and then "*Process/ Edit Input Data*".



From the following dialog select the input file "*FlexSc_FEM.Inp*" and press the "*Select Input File*" button. The menu on the right shows all the data-sets that are saved inside the input file. Select the top batch set: "*Batch for the Flex Satellite with RCS and CMGs*", and click on "*Execute/ View Input Data*".



The Flixan program executes all the data-sets which are called by the batch and saves the spacecraft systems in file "*FlexSc_FEM.Qdr*". It also creates Matlab function m-files of the same systems that can be loaded into Matlab. Click "Exit" to end the Flixan program.



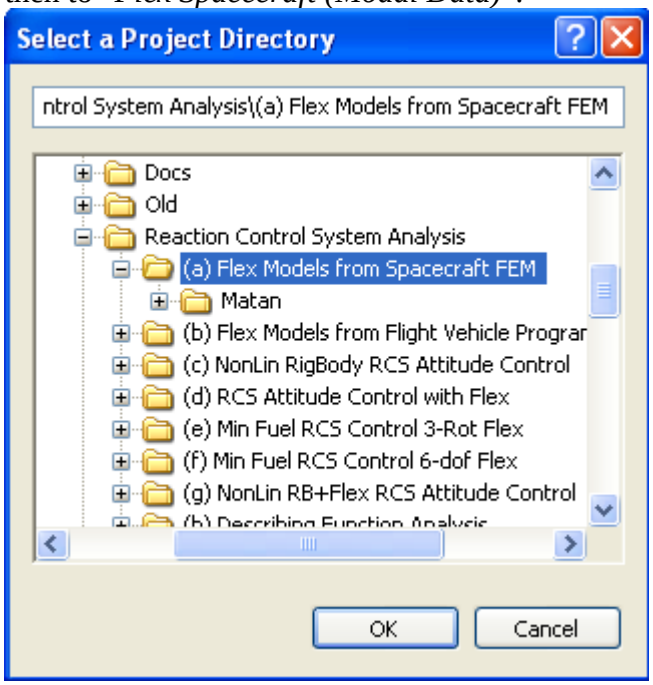
The Matlab function (a,b,c,d) files created are:

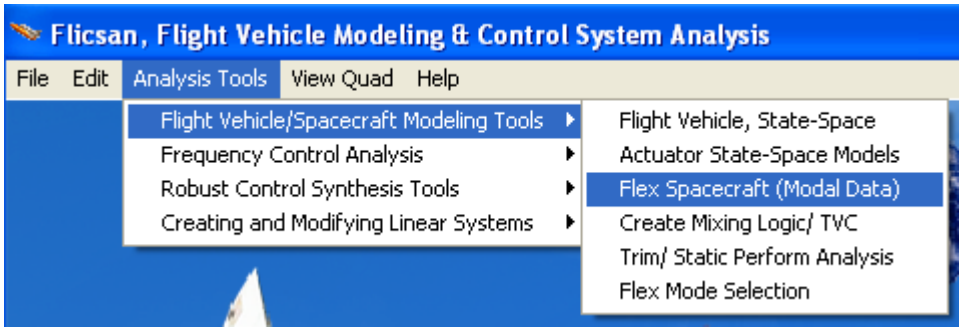
1. (rb_spacecraft_fem_s.m) which contains the system "*Rigid-Body Spacecraft with RCS and CMG*".
2. (rb_spacecraft_fem_z.m) which contains the system "*Rigid-Body Spacecraft with RCS and CMG (Z-Transf)*".
3. (flex_spacecraft_fem_s.m) which contains the system "*RB+Flex Spacecraft with RCS and CMG*".
4. (flex_spacecraft_fem_z.m) which contains the system "*RB+Flex Spacecraft with RCS and CMG (Z-Transf)*".

5. (flex_only_fem_s.m) which contains the system "*Flex Only Spacecraft with RCS and CMG*".
6. (flex_only_fem_z.m) which contains the system "*Flex Only Spacecraft with RCS and CMG (Z-Transf)*".

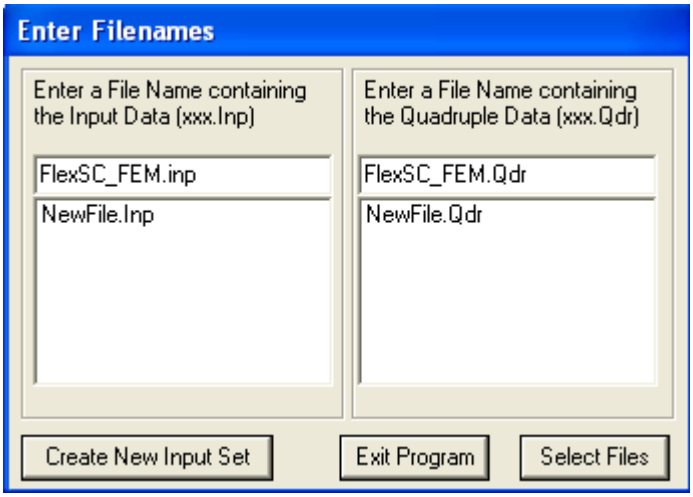
Creating one of the Systems from Scratch

To create one of the systems from scratch, we must first start the Flixan program, then go to folder "*… \Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\ (a) Flex Models from Spacecraft FEM*", go to "*Analysis Tools*", "*Flight Vehicle/ Spacecraft Modeling Tools*", and then to "*Flex Spacecraft (Modal Data)*".

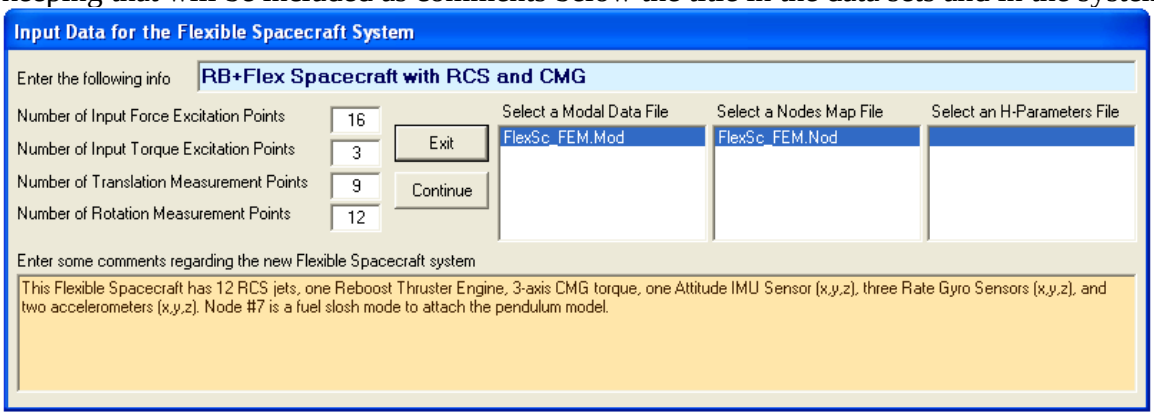




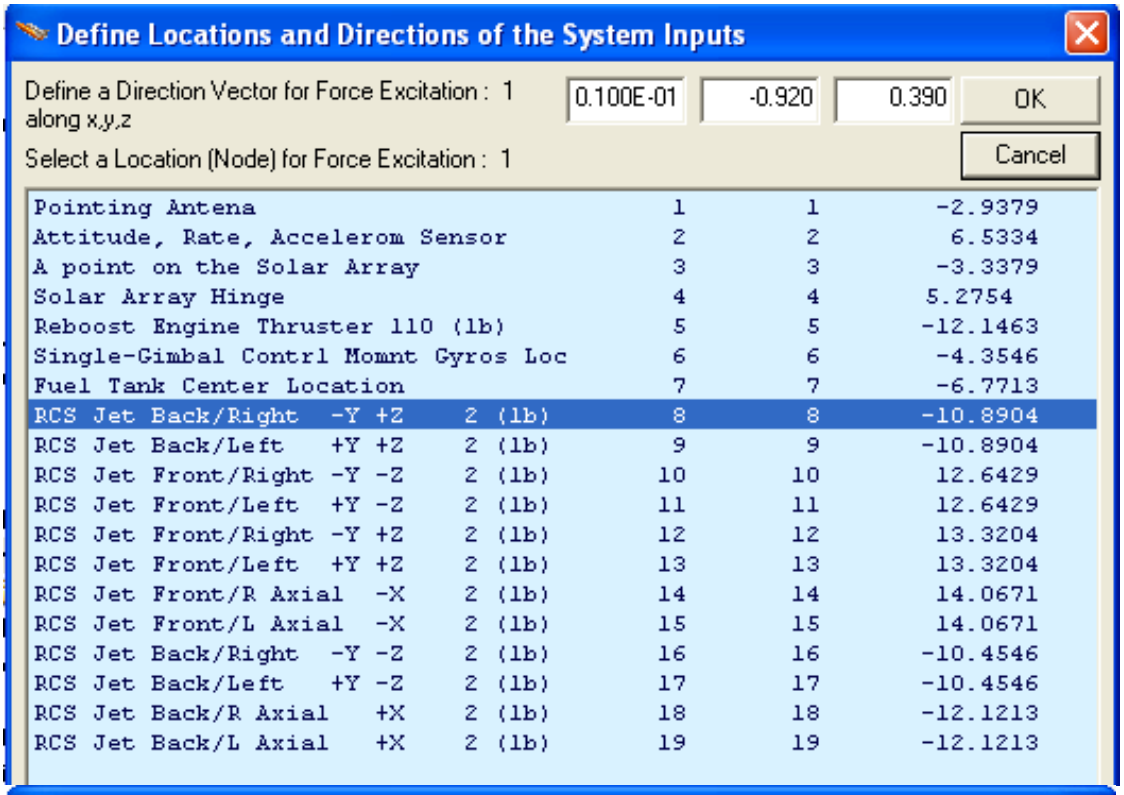
Then we must select the input data file where we are going to save the spacecraft data and output system file that contains the state-space systems. Select or create the files "*FlexSc_FEM.Inp*" and "*FlexSc_FEM.Qdr*", and click on "Create a New Input Set", assuming that we want to create a new set of spacecraft data.

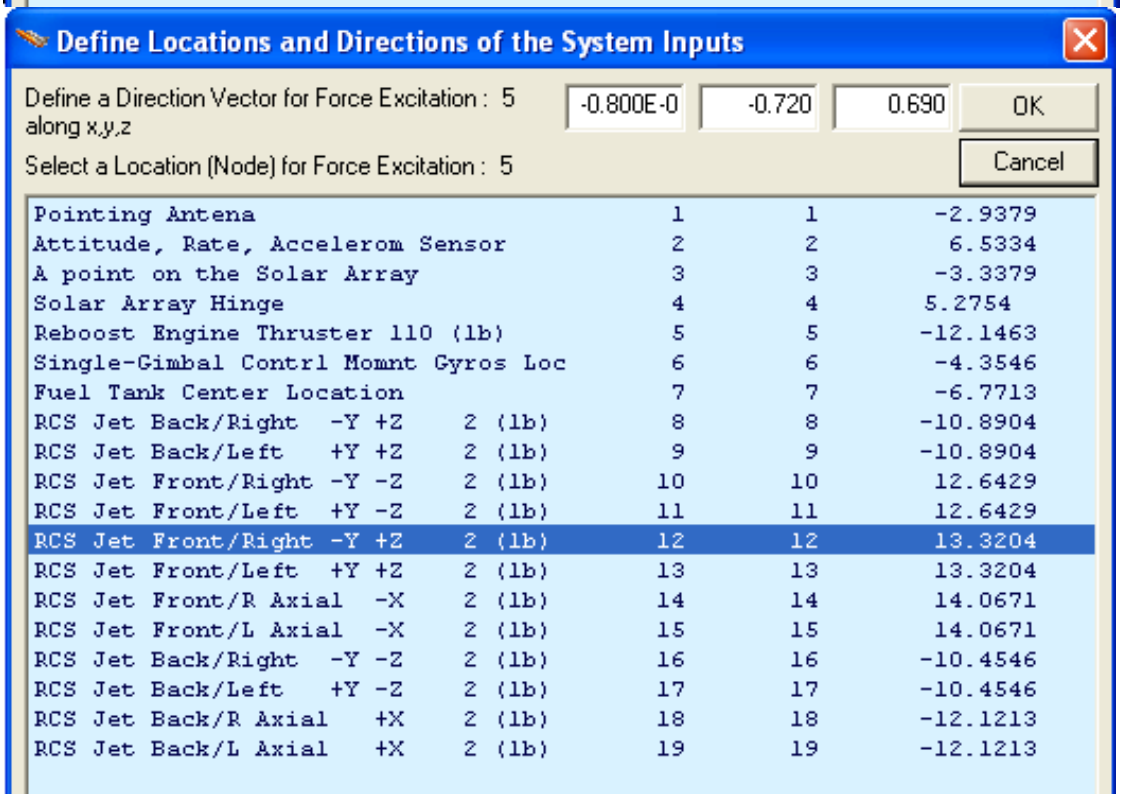


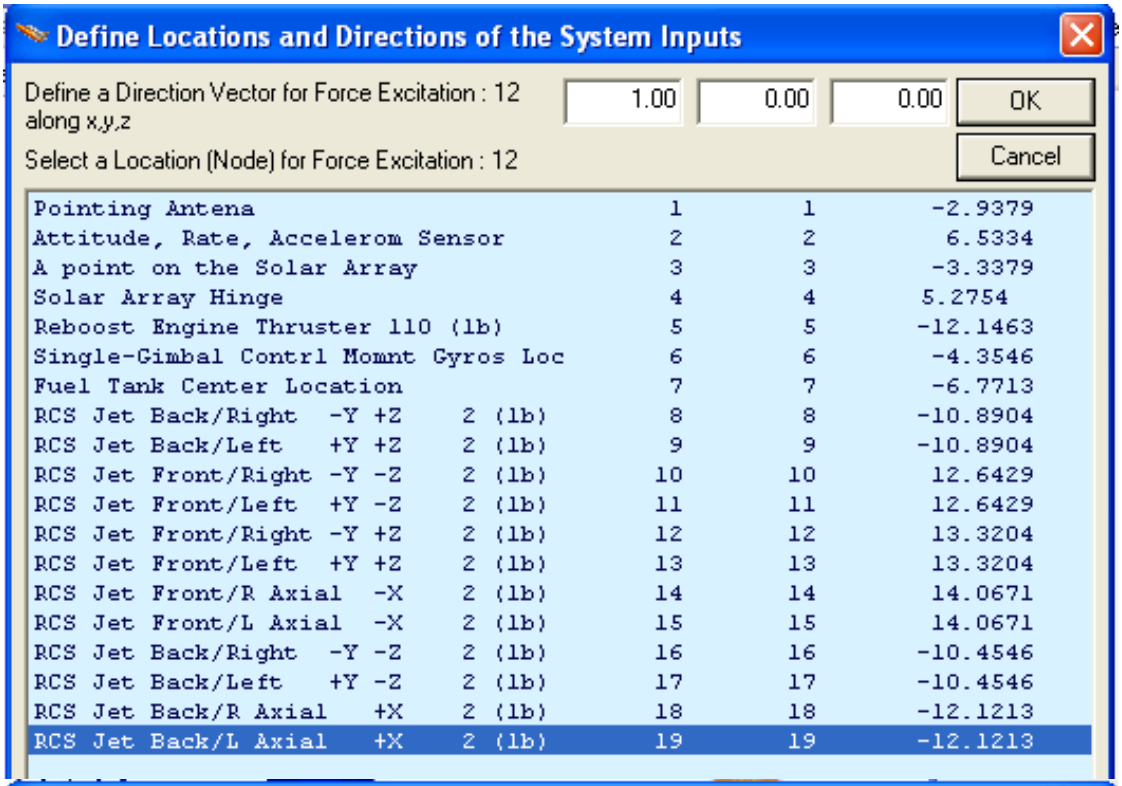
The following dialog is used to define the number of spacecraft excitation and measurement locations. We first define the spacecraft system title "RB+Flex Spacecraft …", 16 force excitation points for RCS jet forces etc, 3 torque excitation points for the CMG (roll, pitch and yaw) torques, 9 translation measurement points for accelerometers, and 12 rotational measurements for ACS gyros and other LOS sensitivity measurement points. We also select the modal data and the nodes filenames "FlexSc_FEM.Mod" and "FlexSc_FEM.Nod", which are already inside the folder. We finally write some notes in the field at the bottom. Notes for our own convenience and book-keeping that will be included as comments below the title in the data sets and in the systems file.

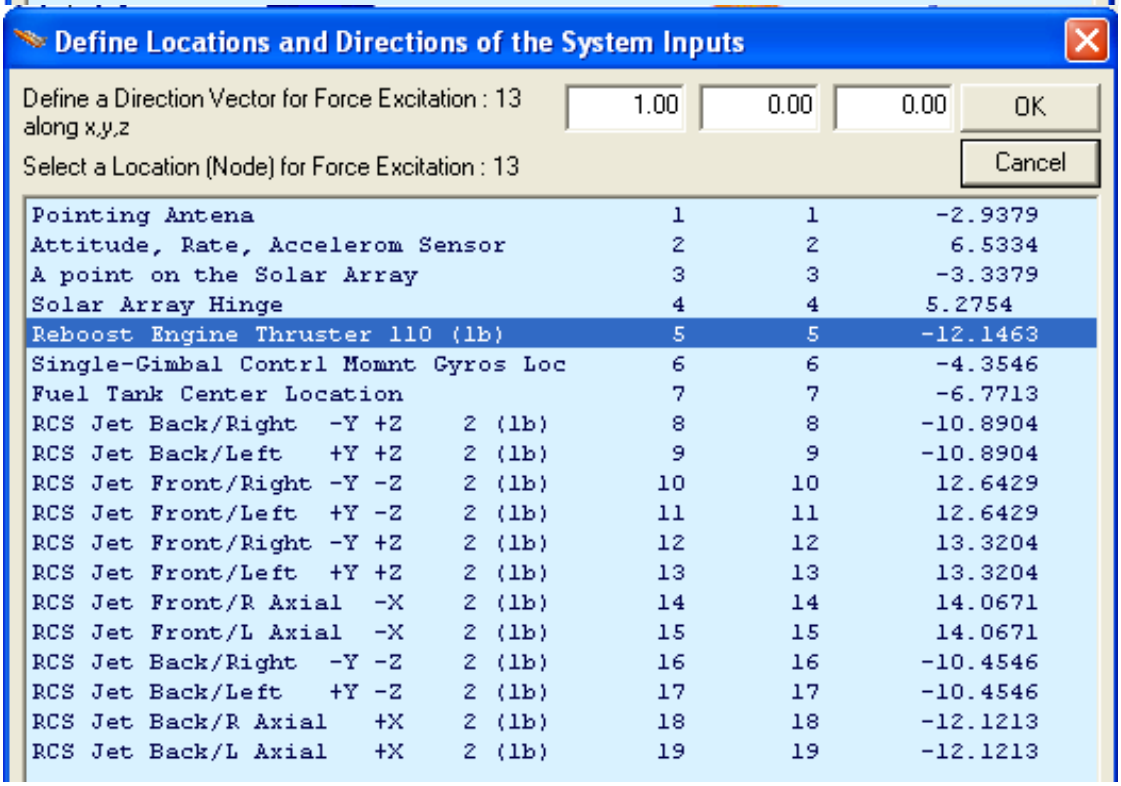


The next step is to define the excitation and measurement points in terms of nodes that correspond to the FEM data and also to define force or measurement directions. The nodes map file appears in a menu form to assist the user in selecting locations for the number of points defined, starting with the force excitations. Force #1 corresponds to RCS jet #1 (Back/ Right –Y +Z) which is at node #8 and its thrust direction unit vector is (x=0.01, y=-0.92, z=0.39). Continue defining the 16 force excitations. Force excitation #5 corresponds to RCS jet #5 (Front/ Right –Y +Z) which is at node #12 and its thrust direction unit vector is (x=0.08, y=-0.72, z=0.69). Similarly we define force #12 to be jet #12 (Back/ L Axial +X) at node #19 with a thrust direction unit vector is (x=1, y=0, z=0). Force #13 is the reboost engine force at node #5 firing in the (1, 0, 0) direction, and so on. Force excitations #14, #15 and #16 are at the tank center node #7, along x, y, and z respectively. Press "OK" every time to continue with the next excitation point.

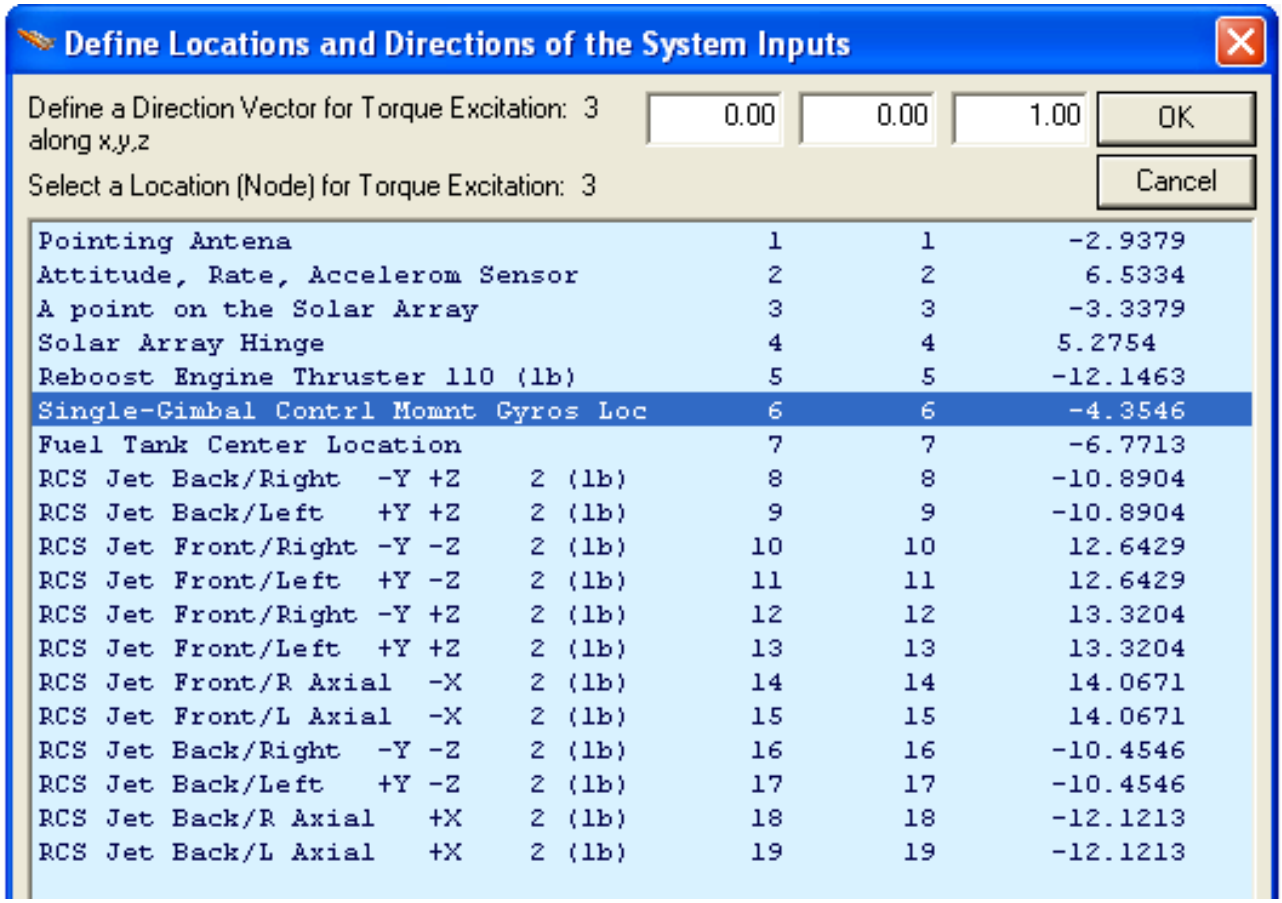

Define Locations and Directions of the System Inputs
Define a Direction Vector for Force Excitation : 1
along x,y,z
0.100E-01
-0.920
0.390
OK
Select a Location (Node) for Force Excitation : 1
Cancel
Pointing Antena 1 1 -2.9379
Attitude, Rate, Accelerom Sensor 2 2 6.5334
A point on the Solar Array 3 3 -3.3379
Solar Array Hinge 4 4 5.2754
Reboost Engine Thruster 110 (lb) 5 5 -12.1463
Single-Gimbal Contrl Momnt Gyros Loc 6 6 -4.3546
Fuel Tank Center Location 7 7 -6.7713
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213


Define Locations and Directions of the System Inputs
Define a Direction Vector for Force Excitation : 5
along x,y,z
-0.800E-0
-0.720
0.690
OK
Select a Location (Node) for Force Excitation : 5
Cancel
Pointing Antena 1 1 -2.9379
Attitude, Rate, Accelerom Sensor 2 2 6.5334
A point on the Solar Array 3 3 -3.3379
Solar Array Hinge 4 4 5.2754
Reboost Engine Thruster 110 (lb) 5 5 -12.1463
Single-Gimbal Contrl Momnt Gyros Loc 6 6 -4.3546
Fuel Tank Center Location 7 7 -6.7713
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213

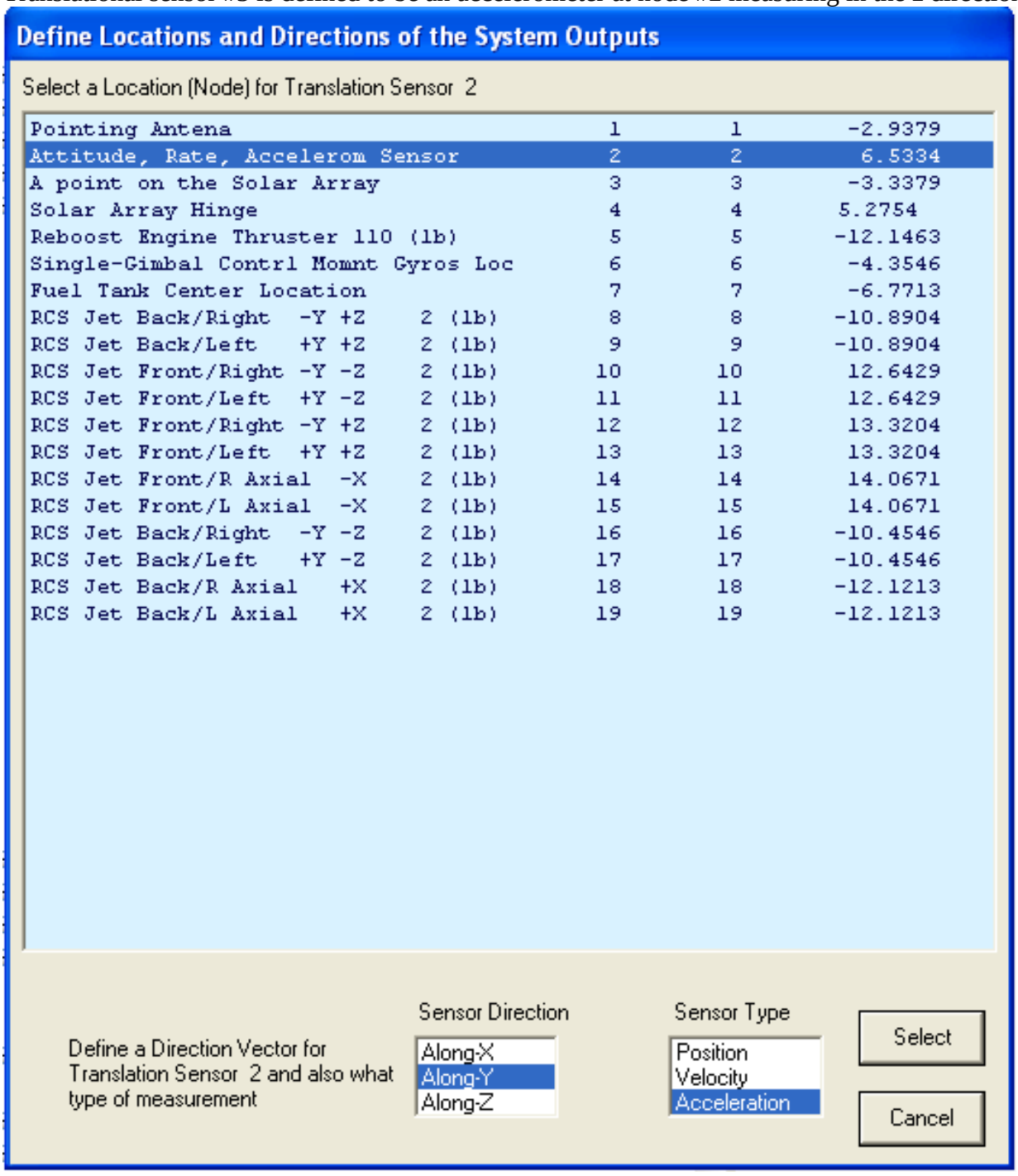

Define Locations and Directions of the System Inputs
Define a Direction Vector for Force Excitation : 12 along x,y,z
1.00
0.00
0.00
OK
Cancel
Select a Location (Node) for Force Excitation : 12
Pointing Antena 1 1 -2.9379
Attitude, Rate, Accelerom Sensor 2 2 6.5334
A point on the Solar Array 3 3 -3.3379
Solar Array Hinge 4 4 5.2754
Reboost Engine Thruster 110 (lb) 5 5 -12.1463
Single-Gimbal Contrl Momnt Gyros Loc 6 6 -4.3546
Fuel Tank Center Location 7 7 -6.7713
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213


Define Locations and Directions of the System Inputs
Define a Direction Vector for Force Excitation : 13 along x,y,z
1.00
0.00
0.00
OK
Cancel
Select a Location (Node) for Force Excitation : 13
Pointing Antena 1 1 -2.9379
Attitude, Rate, Accelerom Sensor 2 2 6.5334
A point on the Solar Array 3 3 -3.3379
Solar Array Hinge 4 4 5.2754
Reboost Engine Thruster 110 (lb) 5 5 -12.1463
Single-Gimbal Contrl Momnt Gyros Loc 6 6 -4.3546
Fuel Tank Center Location 7 7 -6.7713
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213

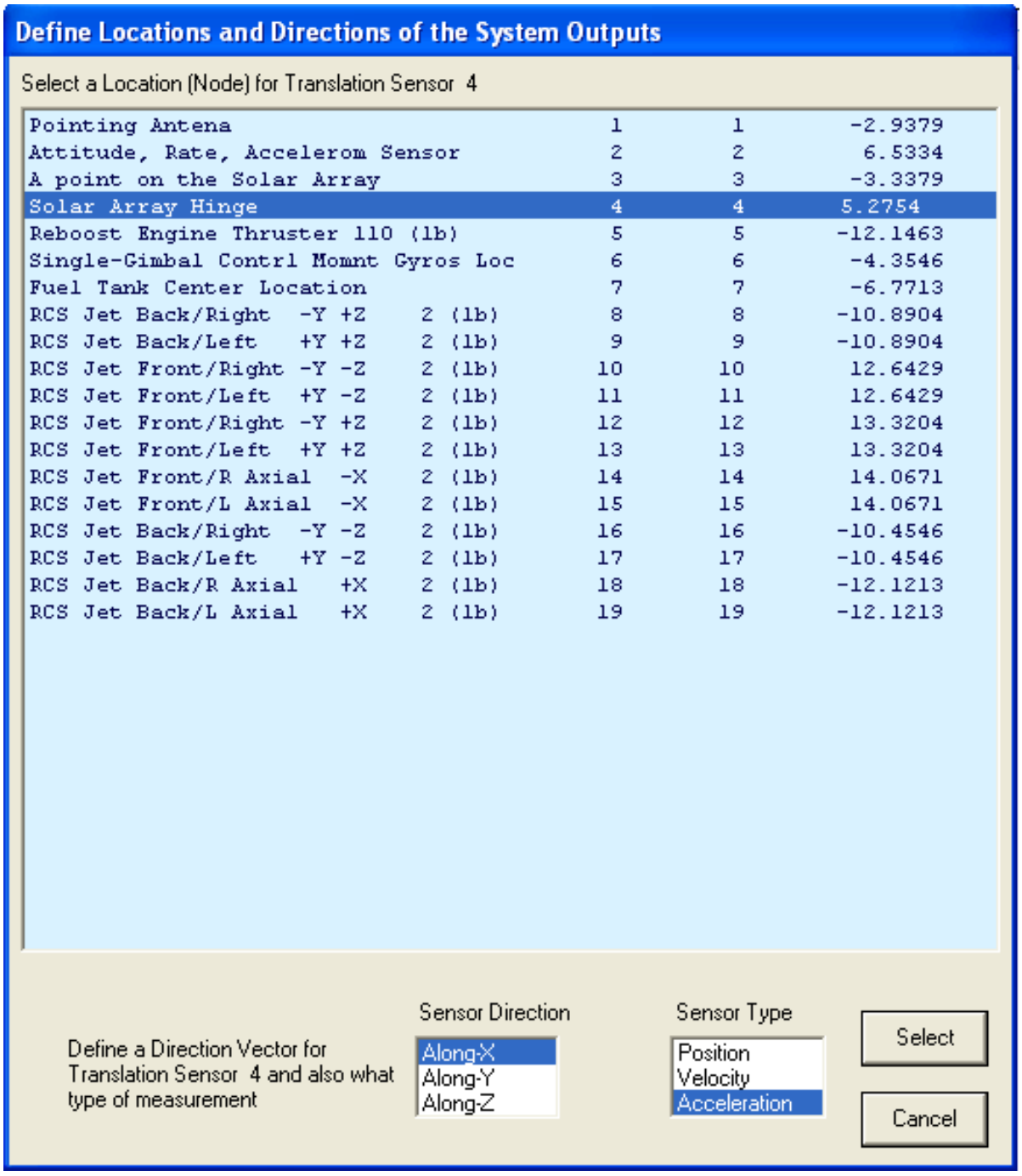
We must also define locations for the 3 torque excitation points define. They correspond to the 3 CMG torques, about the x, y, and z directions. They are all at node #6. For example torque excitation #3, shown below, corresponds to the yaw CMG torque which is at node #6 in the (x=0, y=0, z=1) direction. Press “OK” to define the next point.



The next step is to define the 9 translational measurement points defined. For example, translational sensor #1 is defined to be an accelerometer at node #2 measuring in the x direction. Translational sensor #2, shown below, is defined to be an accelerometer at node #2 measuring in the y direction. Translational sensor #3 is defined to be an accelerometer at node #2 measuring in the z direction.

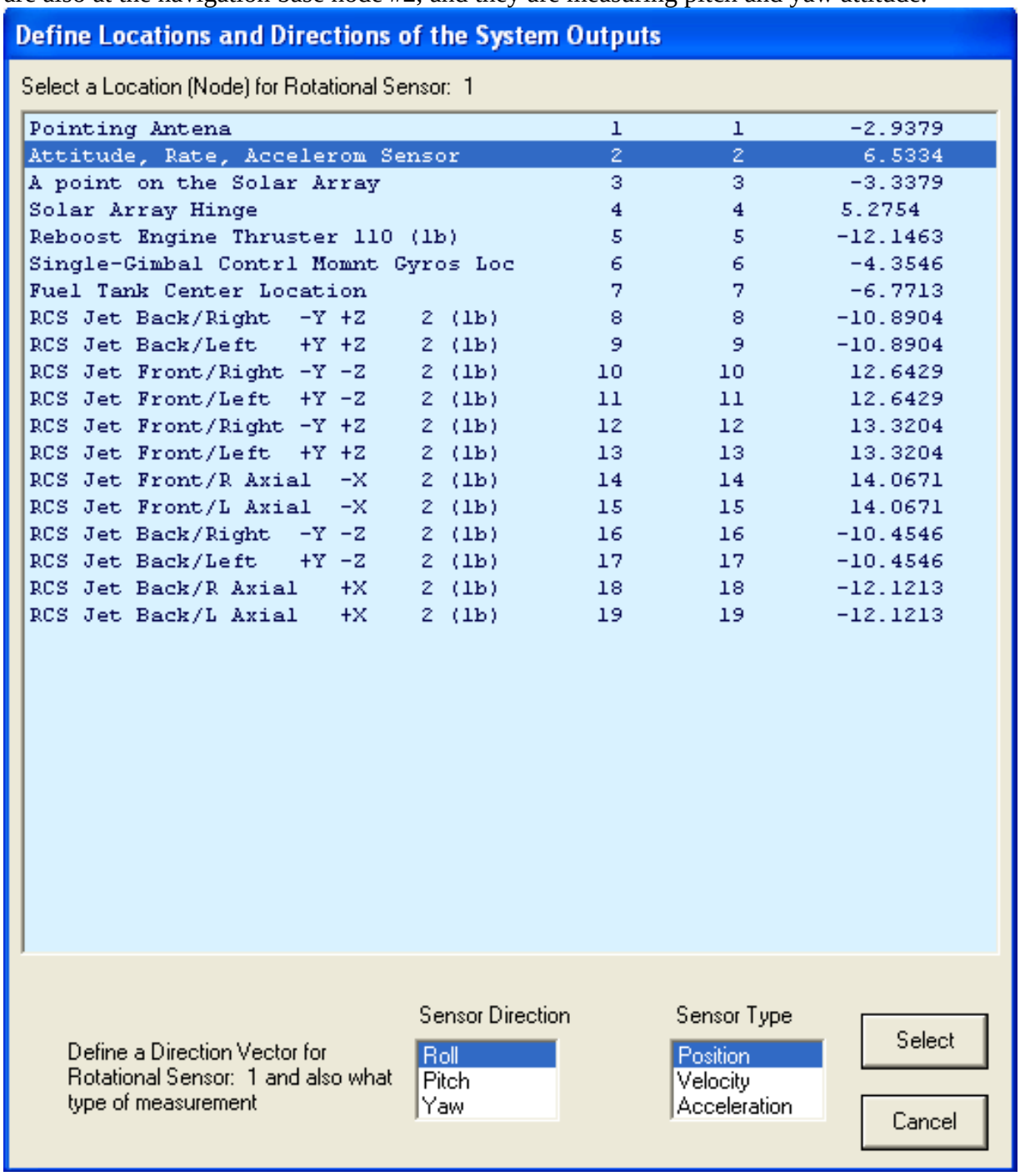


Similarly, the translation sensors #4, #5, and #6 are at node #4, the solar array hinge, measuring accelerations along the x, y, and z axes.

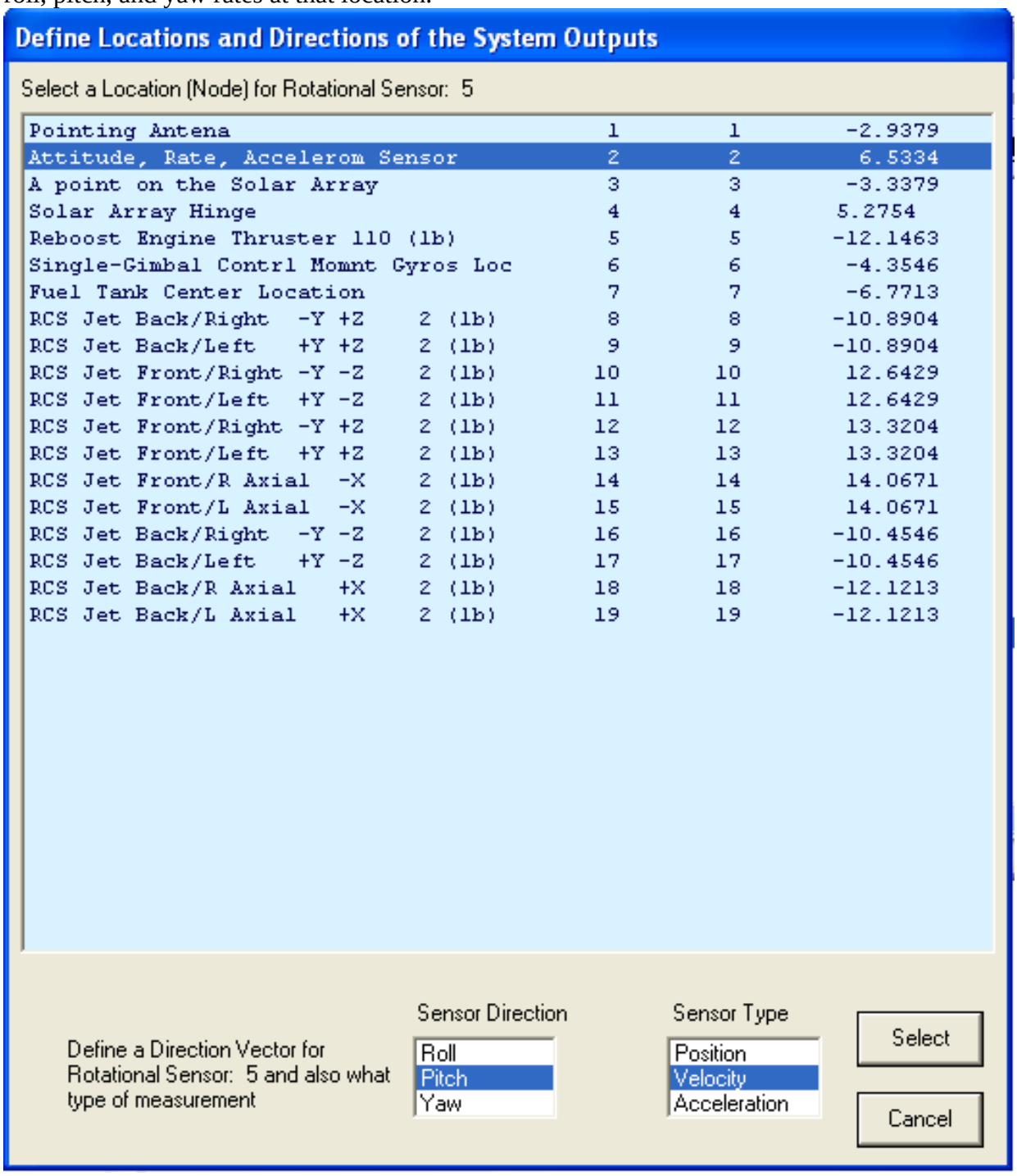


Similarly, the translation sensors #7, #8, and #9 are at node #7, the fuel tank center, measuring accelerations along the x, y, and z axes. These locations are needed for coupling the spacecraft with the slosh model.

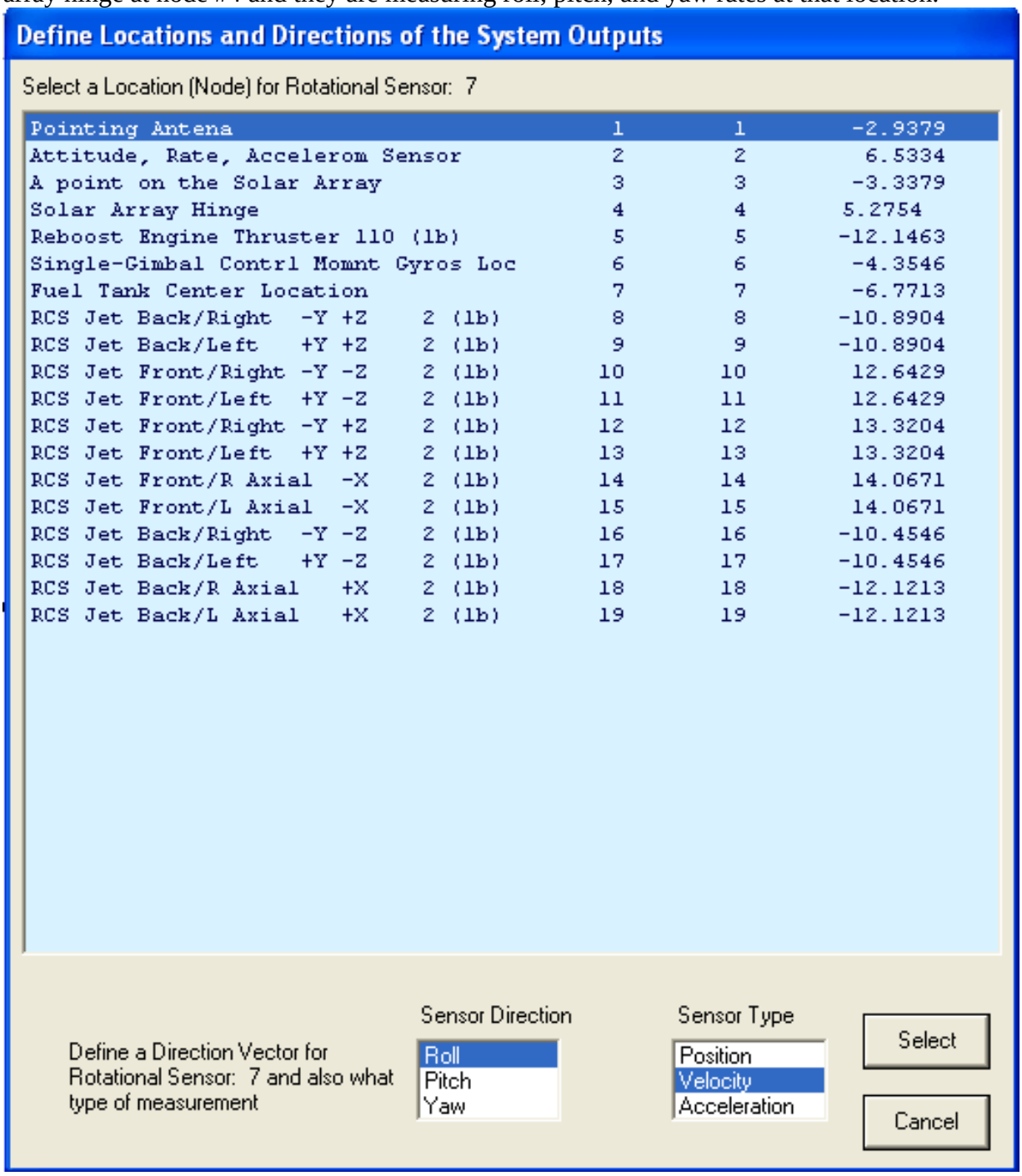
Finally, we must define the 12 rotational sensor points. Rotational sensor #1, see below, is at the navigation base at node #2 and it is measuring roll attitude. Similarly, rotational sensors #2 and #3 are also at the navigation base node #2, and they are measuring pitch and yaw attitude.



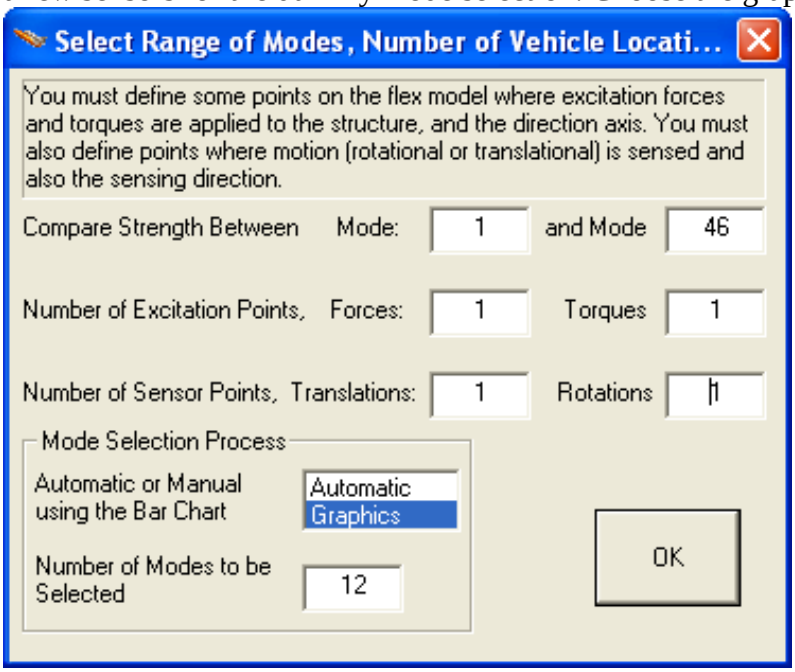
Rotational sensors #4, #5 and #6 are also at the navigation base at node #2 and they are measuring roll, pitch, and yaw rates at that location.

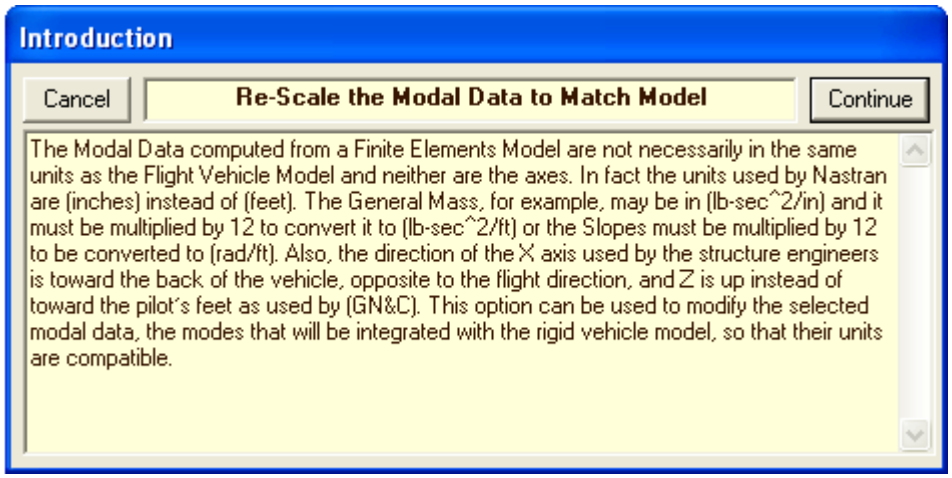


Rotational sensors #7, #8 and #9 are at the pointing antenna at node #1 and they are measuring roll, pitch, and yaw rates at that location. Similarly, rotational sensors #10, #11 and #12 are at the solar array hinge at node #4 and they are measuring roll, pitch, and yaw rates at that location.

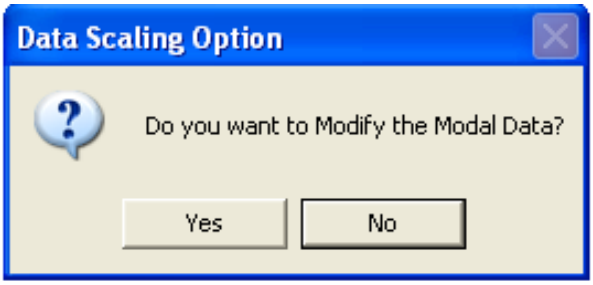


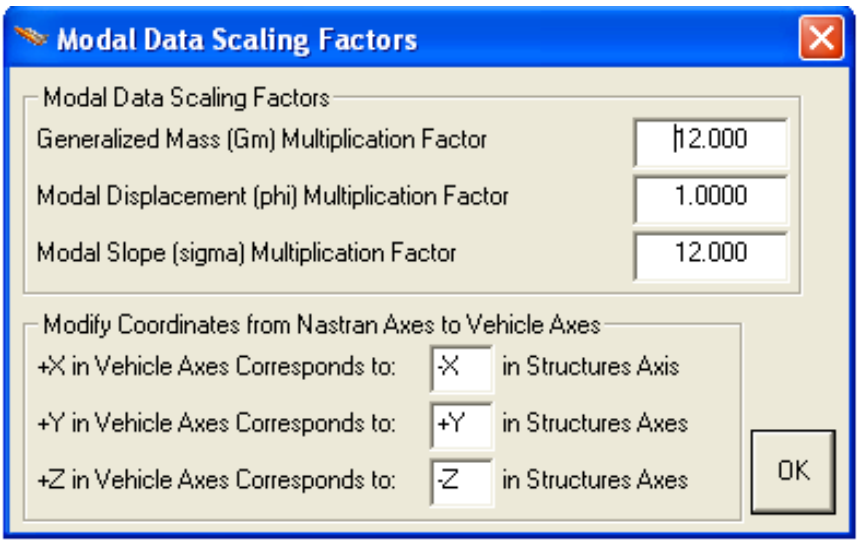
Now that we have defined the excitation and measurement points on the spacecraft the next step is to select some of the flex modes. The program is now ready to perform a mode selection operation, and to choose or allow the user to choose some of the most dominant flex modes. This operation, however, is not important for us right now because we want to include all 46 modes in our model, but we still have to go through a dummy selection process. This operation will also rescale the modal data which are in units of inches originally. From the following menu we choose to compare the modal strength of all modes (from 1 to 46), and we must also choose a few excitation points and a few sensors for the dummy mode selection. Choose the graphic option, and click "OK".



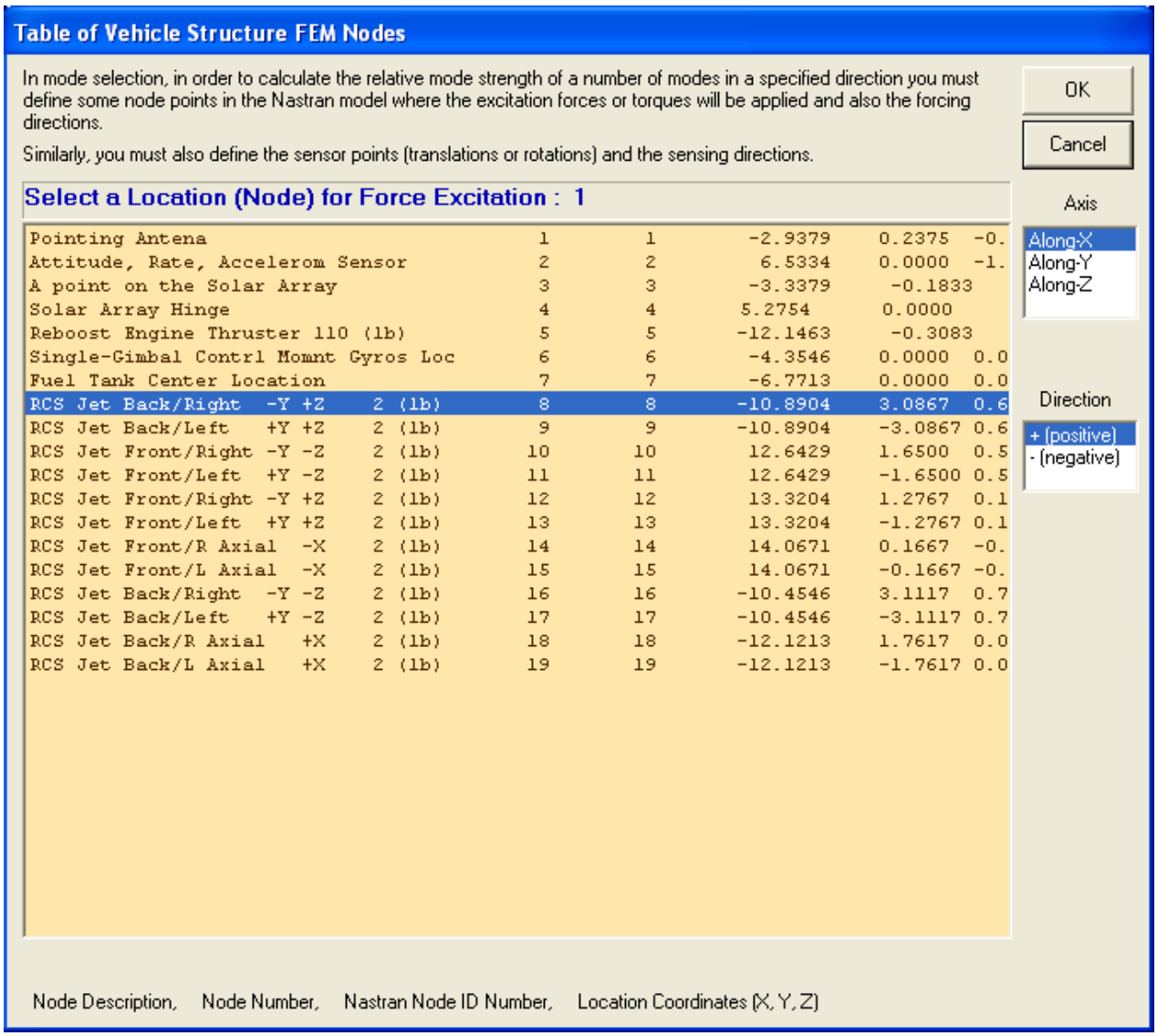


In the following question dialog answer "Yes" to modify the data, and in the next dialog keep the defaults which will rescale the system from units of (inches) to units of (feet). It will also reverse the x and z directions because we would like to have the x direction towards the LOS and the z down, in contrast to structures who prefer the opposite.

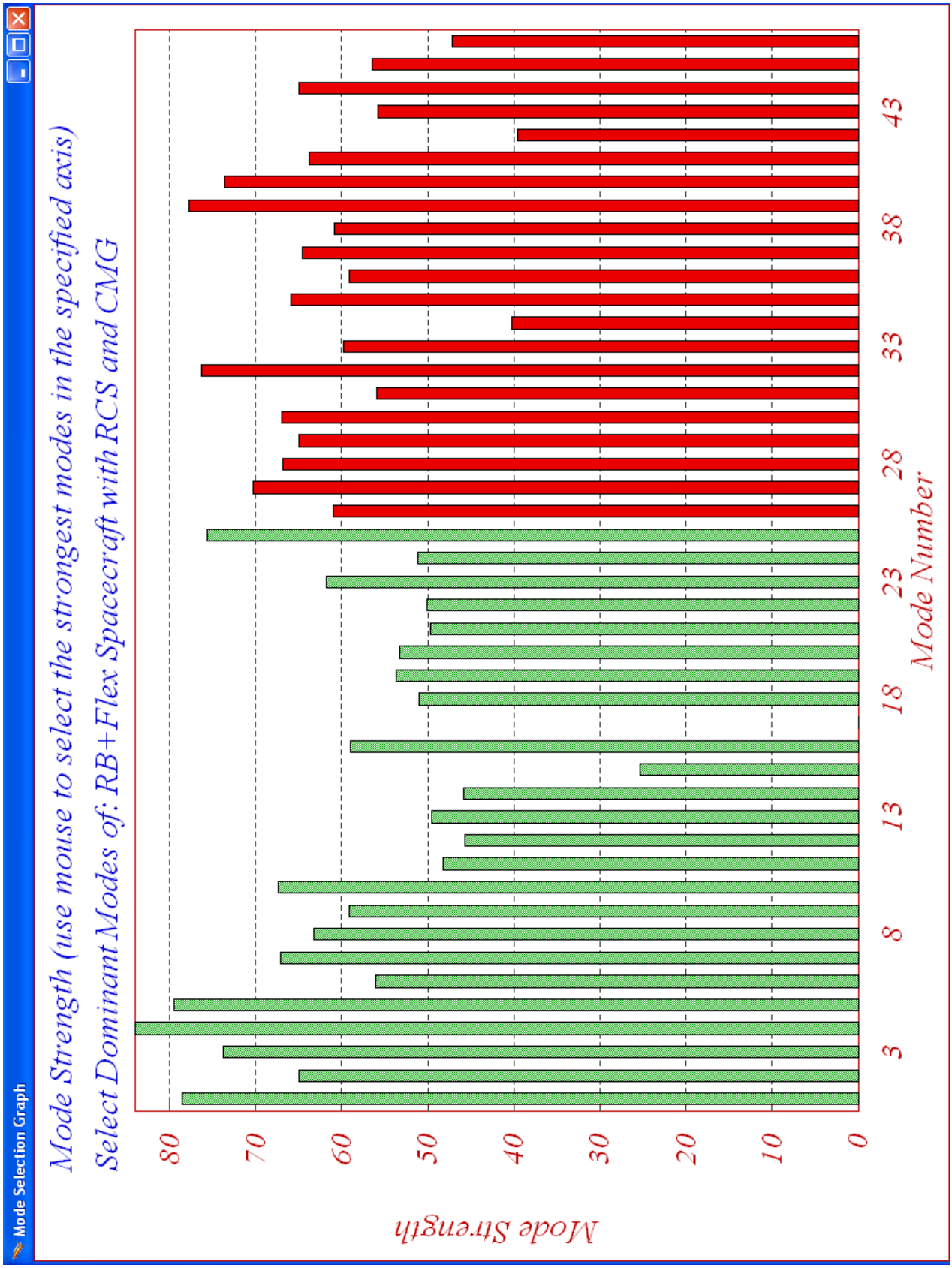




We finally select some dummy force and torque excitation points and some dummy translational and rotational measurement points required by the program in the mode selection process as it calculates the modal strengths. As we said earlier, we don't care in this case which points to pick because we have already decided to include all the modes. For a force excitation, for example, we can pick the first RCS jet at node #8 firing in any direction. For rotational measurement we may pick the ACS gyros at node #2 measuring in any direction. But we must be careful in selecting the excitation and measurement points because there are also zeros in the modal data, because some nodes have only rotations or only translations defined. In this case there will be a zero transfer between the excitation and measurement points and the program will terminate without mode selection results.

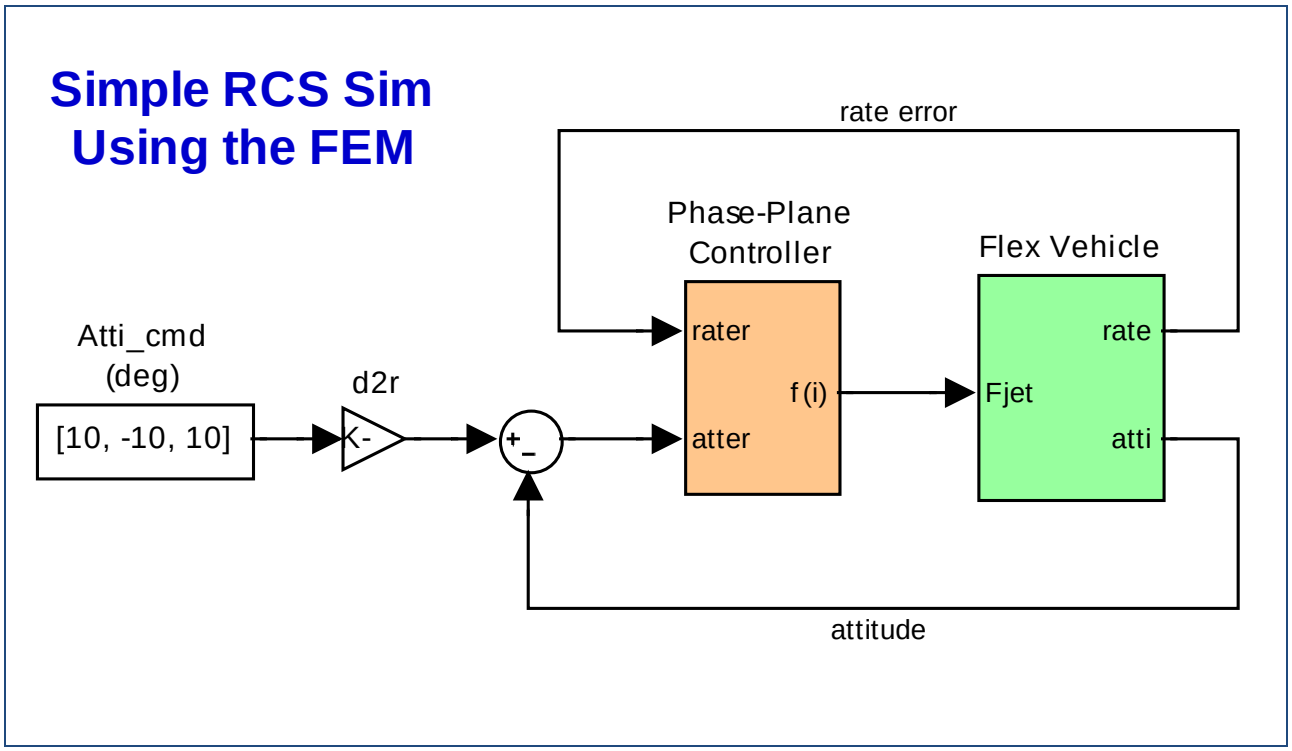


The program finally calculates the modal strengths for the 46 modes and displays it in bar-chart. Each bar corresponds to a mode number. The user must select all the modes from the chart, shown below, by clicking on the bar with the mouse. When a mode is selected its color changes from red to green.

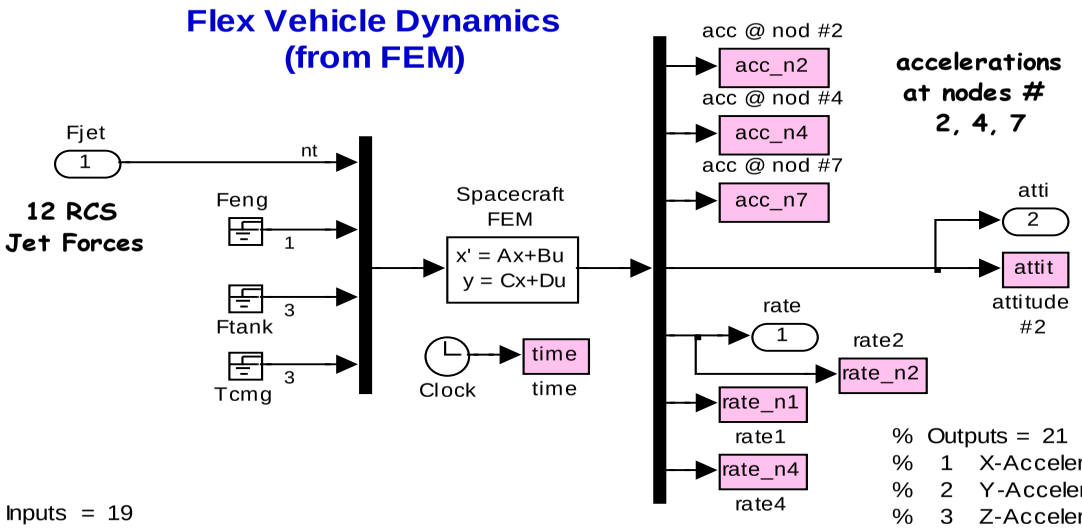

Mode Selection Graph
Mode Strength (use mouse to select the strongest modes in the specified axis)
Select Dominant Modes of: RB+Flex Spacecraft with RCS and CMG
Mode Strength
0
10
20
30
40
50
60
70
80
3
8
13
18
23
28
33
38
43
Mode Number

Simple Simulation Using the Finite Element Model

The following simulation model "*Flex_Sim.mdl*" uses the system title "*RB+Flex Spacecraft with RCS and CMG*", which was created using modal data as described earlier. It is saved in folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\(a) Flex Models from Spacecraft FEM\Matan*".



The simulation consists of the spacecraft dynamics and a simple phase-plane controller "*Phase-Plane.m*". It is initialized by file "*start.m*" which loads the spacecraft state-space system from file "*flex_spacecraft_fem_s.m*". The rigid-body modes are included in the finite element model. The following figure shows the spacecraft attitude response to 10 (deg) attitude commands in different directions. It also shows accelerometer and rate gyro responses in three spacecraft locations. This is a simple model to begin. We will continue with more complex models in the following sections.



% Inputs = 19
% 1 Force No 1 Applied at Node # 8 (lbf)
% 2 Force No 2 Applied at Node # 9 (lbf)
% 3 Force No 3 Applied at Node # 10 (lbf)
% 4 Force No 4 Applied at Node # 11 (lbf)
% 5 Force No 5 Applied at Node # 12 (lbf)
% 6 Force No 6 Applied at Node # 13 (lbf)
% 7 Force No 7 Applied at Node # 14 (lbf)
% 8 Force No 8 Applied at Node # 15 (lbf)
% 9 Force No 9 Applied at Node # 16 (lbf)
% 10 Force No 10 Applied at Node # 17 (lbf)
% 11 Force No 11 Applied at Node # 18 (lbf)
% 12 Force No 12 Applied at Node # 19 (lbf)

% 13 Force No 13 Applied at Node # 5 (lbf)

% 14 Force No 14 Applied at Node # 7 (lbf)
% 15 Force No 15 Applied at Node # 7 (lbf)
% 16 Force No 16 Applied at Node # 7 (lbf)

% 17 Torque No 1 Applied at Node # 6 (ft-lb)
% 18 Torque No 2 Applied at Node # 6 (ft-lb)
% 19 Torque No 3 Applied at Node # 6 (ft-lb)

% Outputs = 21
% 1 X-Accelerat. Sensed at Node # 2 (ft)
% 2 Y-Accelerat. Sensed at Node # 2 (ft)
% 3 Z-Accelerat. Sensed at Node # 2 (ft)

% 4 X-Accelerat. Sensed at Node # 4 (ft)
% 5 Y-Accelerat. Sensed at Node # 4 (ft)
% 6 Z-Accelerat. Sensed at Node # 4 (ft)

% 7 X-Accelerat. Sensed at Node # 7 (ft)
% 8 Y-Accelerat. Sensed at Node # 7 (ft)
% 9 Z-Accelerat. Sensed at Node # 7 (ft)

% 10 X-Rotation Sensed at Node # 2 (radian)
% 11 Y-Rotation Sensed at Node # 2 (radian)
% 12 Z-Rotation Sensed at Node # 2 (radian)

% 13 X-Rot. Rate Sensed at Node # 2 (radian)
% 14 Y-Rot. Rate Sensed at Node # 2 (radian)
% 15 Z-Rot. Rate Sensed at Node # 2 (radian)

% 16 X-Rot. Rate Sensed at Node # 1 (radian)
% 17 Y-Rot. Rate Sensed at Node # 1 (radian)
% 18 Z-Rot. Rate Sensed at Node # 1 (radian)

% 19 X-Rot. Rate Sensed at Node # 4 (radian)
% 20 Y-Rot. Rate Sensed at Node # 4 (radian)
% 21 Z-Rot. Rate Sensed at Node # 4 (radian)

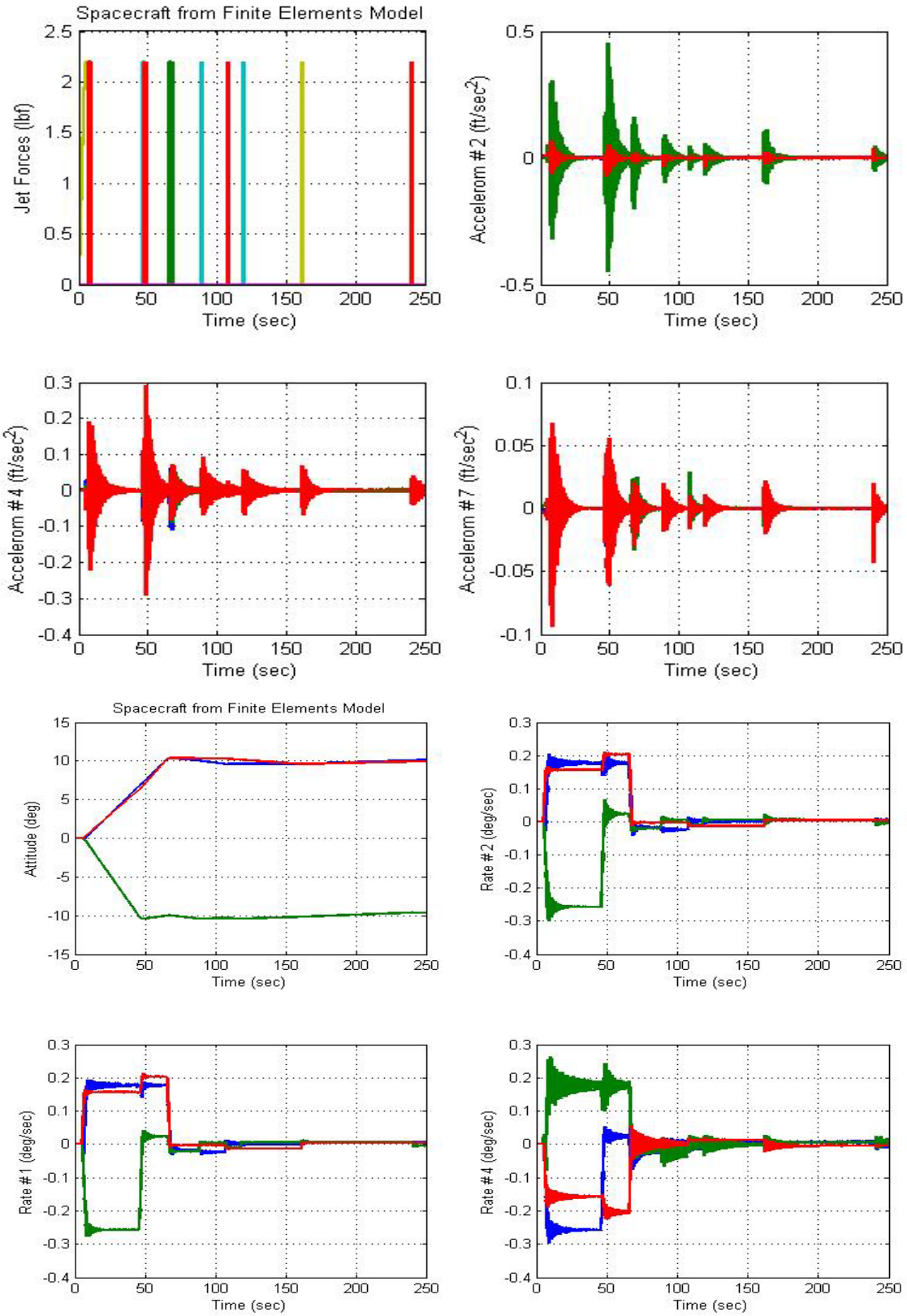


Figure 1.1.1 Flexible spacecraft response to 10 (deg) attitude commands.

Frequency Domain Stability Analysis

In the same folder there is also a Simulink model "*Open_Loop_RCS.mdl*" used for analyzing RCS flex mode stability using the Describing Function (DF) method.

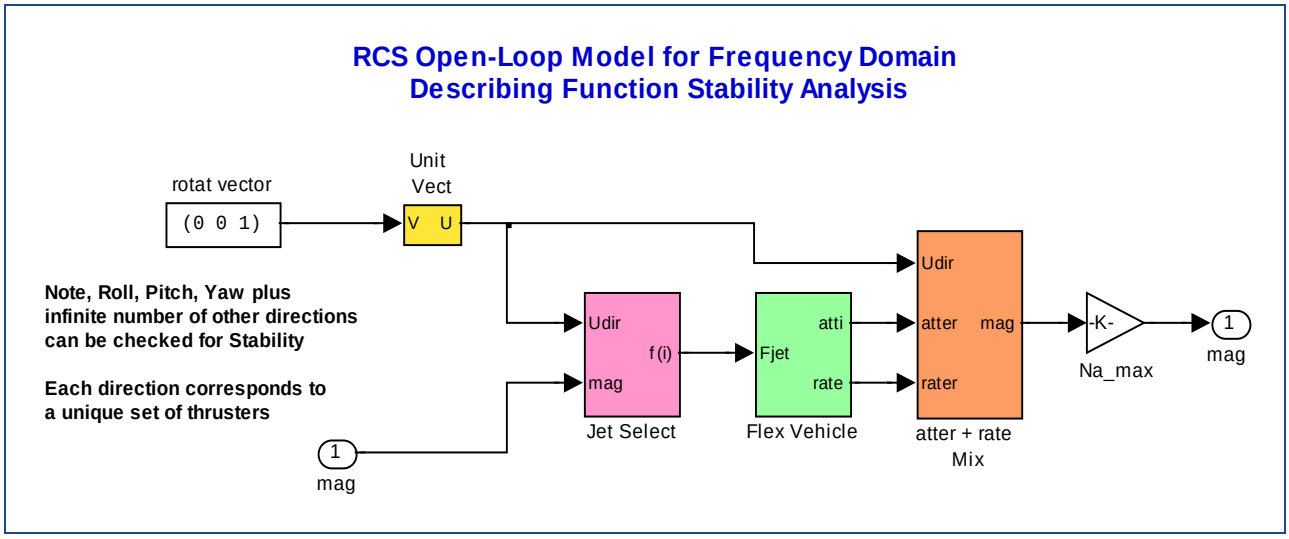


Figure 1.1.2 Frequency domain model for RCS stability analysis using Describing Function

This model is an open-loop linearized version of the previous closed-loop model. It consists of the same basic subsystems, slightly modified for frequency domain analysis. Stability is evaluated one axis (rotational direction) at a time. The rotational direction is an input to the model. In the example above we are analyzing the yaw axis. By modifying the rotation vector we can analyze pitch (0, 1, 0) and roll (1, 0, 0). It is also possible to analyze stability in many other skewed directions such as: (0, 0.7, -0.7) or (0.5, -0.3, 0.6), since every direction corresponds to a unique set of thrusters exciting the flex modes differently. For linear analysis the complex non-linear phase-plane logic is approximated with a linear combination of attitude plus rate errors. The linearized jet selection logic is also an approximation because it averages the positive and negative accelerations about a specific rotational vector. It selects not only the positive direction jets (with half thrust) but also the jets that accelerate in the opposite direction but assuming negative half thrusts, thus, exercising both positive and negative jets. The system output is multiplied by the max value of the dead-band DF to scale the Nichols charts so that the Nichols critical point (+) corresponds to the min of the DF inverse, that is -1/N(a). The Matlab file "frequ.m" uses the above model to calculate the frequency response and plot the Nichols charts as shown in Figure (1.1.3) below.

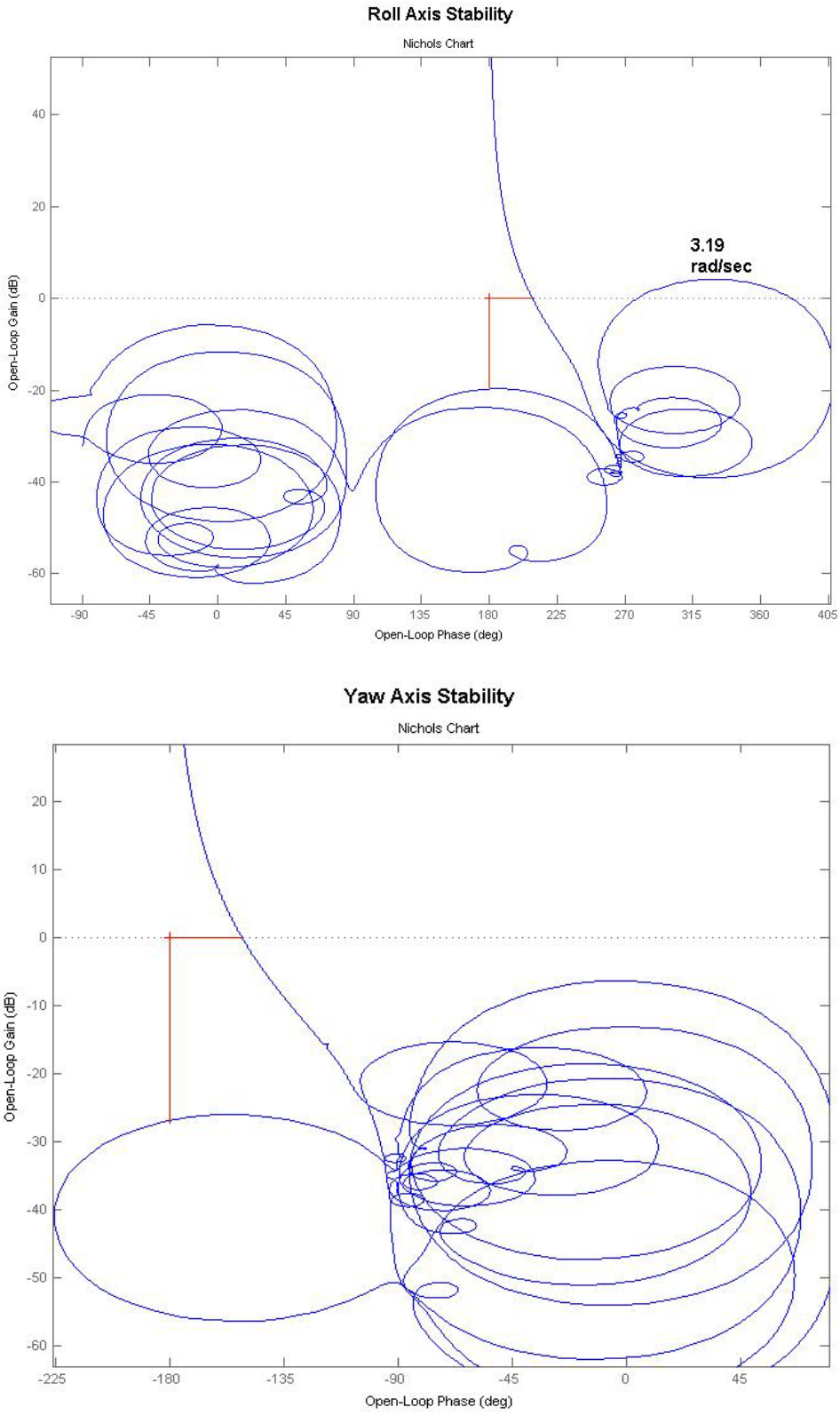
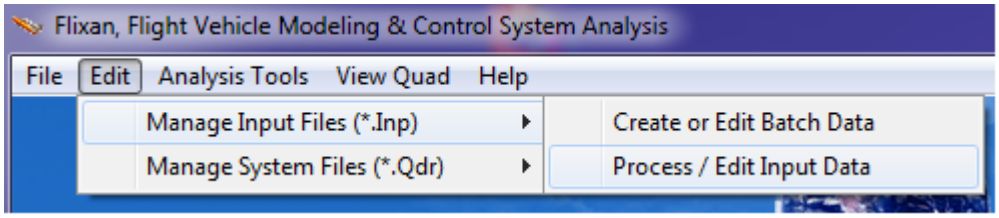


Figure 1.1.3 Nichols Charts showing stability margins from the minimum point of -1/N(a)

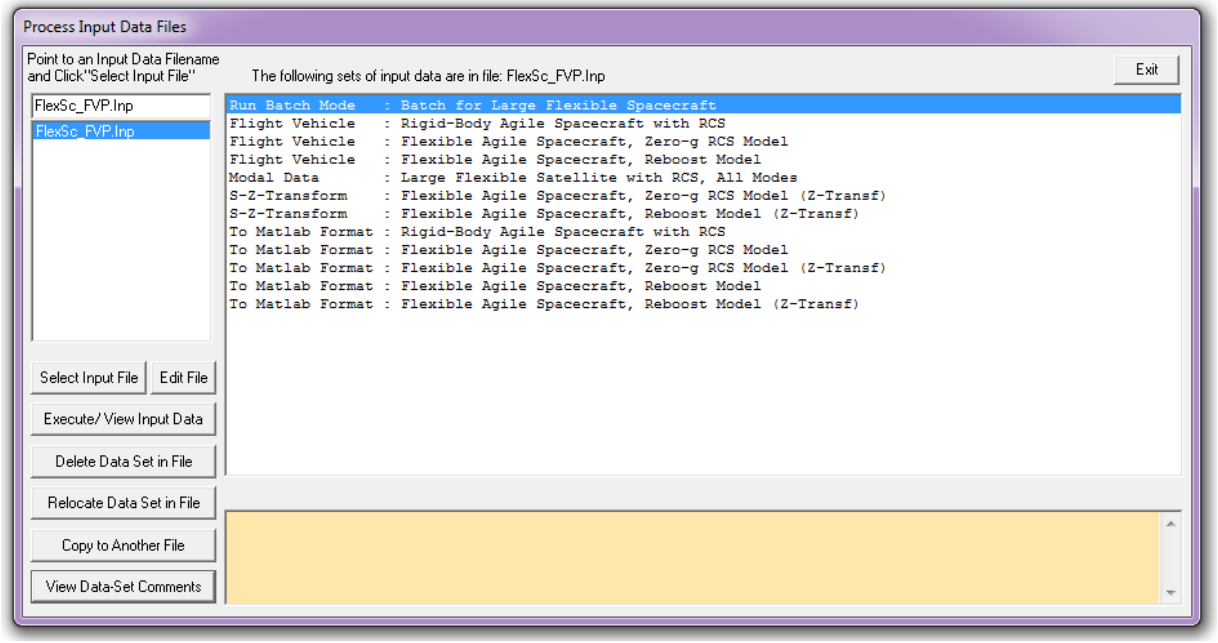
1.2 Models Created using the Flight Vehicle Modeling Program

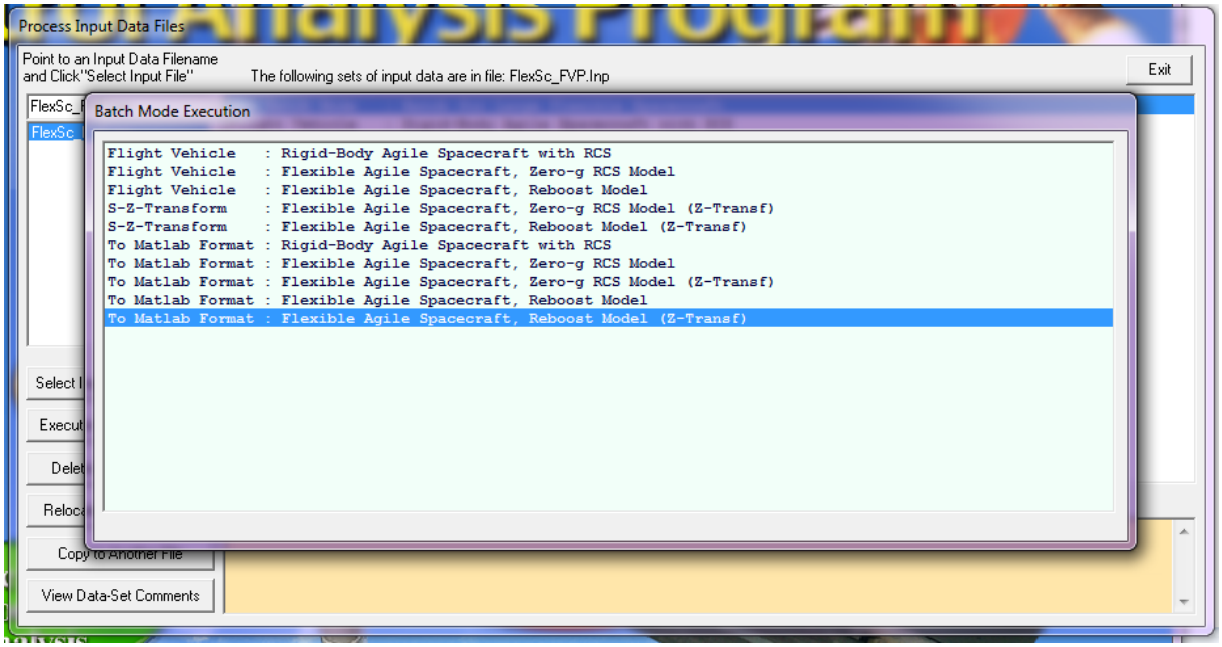
Flexible spacecraft models can also be created using the Flight Vehicle Modeling Program (FVP). The following analysis is similar to Section 1.1 except that it uses the FVP to model the same spacecraft. The input data and the analysis are saved in a separate folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\(b) Flex Models from Flight Vehicle Program*". Take a look at the input data file "*FlexSc_FVP.Inp*". This file creates 3 flight vehicle models: a rigid-body model "*Rigid-Body Agile Spacecraft with RCS*", a flexible spacecraft model for zero g RCS analysis "*Flexible Agile Spacecraft, Zero-g RCS Model*", and an accelerating flex model for RCS control during re-boost when the main engine is firing "*Flexible Agile Spacecraft, Reboost Model*". Some of these spacecraft models are z-transformed using 5 msec sampling period for discrete analysis. Their titles are slightly modified, extended with the label "(Z-Transf)". The continuous and discrete systems are then re-formatted for Matlab analysis. The Matlab m-function state-space system files for the zero g models are: "*Flex_Spacecraft_fvp_s*" and "*Flex_Spacecraft_fvp_z*", and for the accelerating models are: "*Reboost_fvp_s*" and "*Reboost_fvp_z*". These systems are loaded into Matlab workspace. The systems file "*FlexSc_FVP.Qdr*" contains the state-space systems with comments and input/ output definitions.

There is a batch set on the top of the input data file "*FlexSc_FVP.Inp*". Its title is "*Batch for Large Flexible Spacecraft*", and it is used to speed up the data processing by the Flixan program. To execute this batch, start Flixan, go to "*Edit*", select "*Manage Input Files (*.Inp)*", and "*Process/ Edit Input Data*".



In the following dialog there is a menu on the left side listing the input data files in the selected directory. In this case there is only one file "*FlexSc_FVP.Inp*". Click on the file name and then click on "*Select Input File*". The menu on the right side of the dialog will display the titles of all the data sets which are saved in the input file. The program which processes the data-set appears on the left of the title. The creation of the data sets is not shown in this example because it has been demonstrated in other similar examples. Select the first set which is the batch that runs all the other sets, and click on "*Execute/ View Input Data*". The batch executes each data set sequentially and saves the data in file "*FlexSc_FVP.Qdr*".

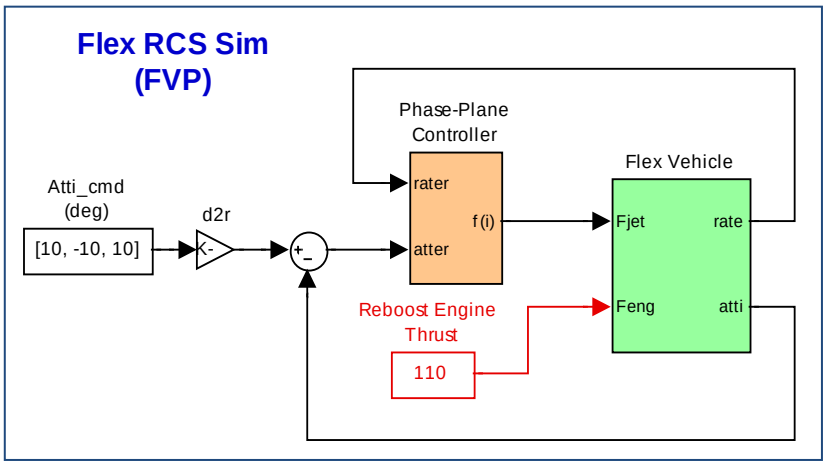




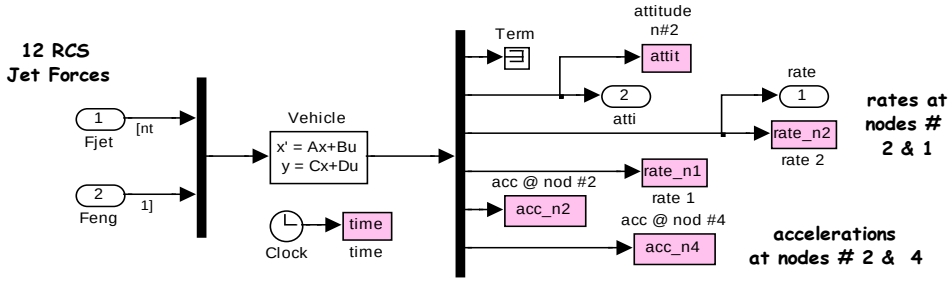
The input data file contains also a set of selected modal data. Its title is "*Large Flexible Satellite with RCS, All Modes*". The modal data set contains 40 pre-selected and pre-processed modes which is used by the FVP to model structural flexibility. Only flex modes were selected, no rigid-body modes since the rigid-body dynamics is provided by the FVP. The mode selection and preparation step is not shown in this example because it has been demonstrated in other examples. The modal data file "*FlexSc_FEM.Mod*" and the nodes map "*FlexSc_FEM.Mod*" were used during the mode selection process. The modal data consists of 40 sets of data, a set for each of the selected modes. The data consist of mode frequency and damping, and the mode shapes, translations and rotations at key locations, such as, the 12 RCS jets, the 9 gyros (3 locations times 3 directions each), the 6 accelerometers (2 locations x 3 directions each), the center of the fuel tank, and the reboost engine.

Simple Simulation that uses the FVP Model

The following simulation model "*Flex_Sim.mdl*" uses the system title "*Flexible Agile Spacecraft, Zero-g RCS Model*", which was created using the Flight Vehicle Modeling Program as described earlier. It is saved in folder "*C:\Flixan\ Examples\ Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\(b) Flex Models from Flight Vehicle Program\Matan*".



Flex Vehicle Dynamics from Flight Vehicle Modeling Program

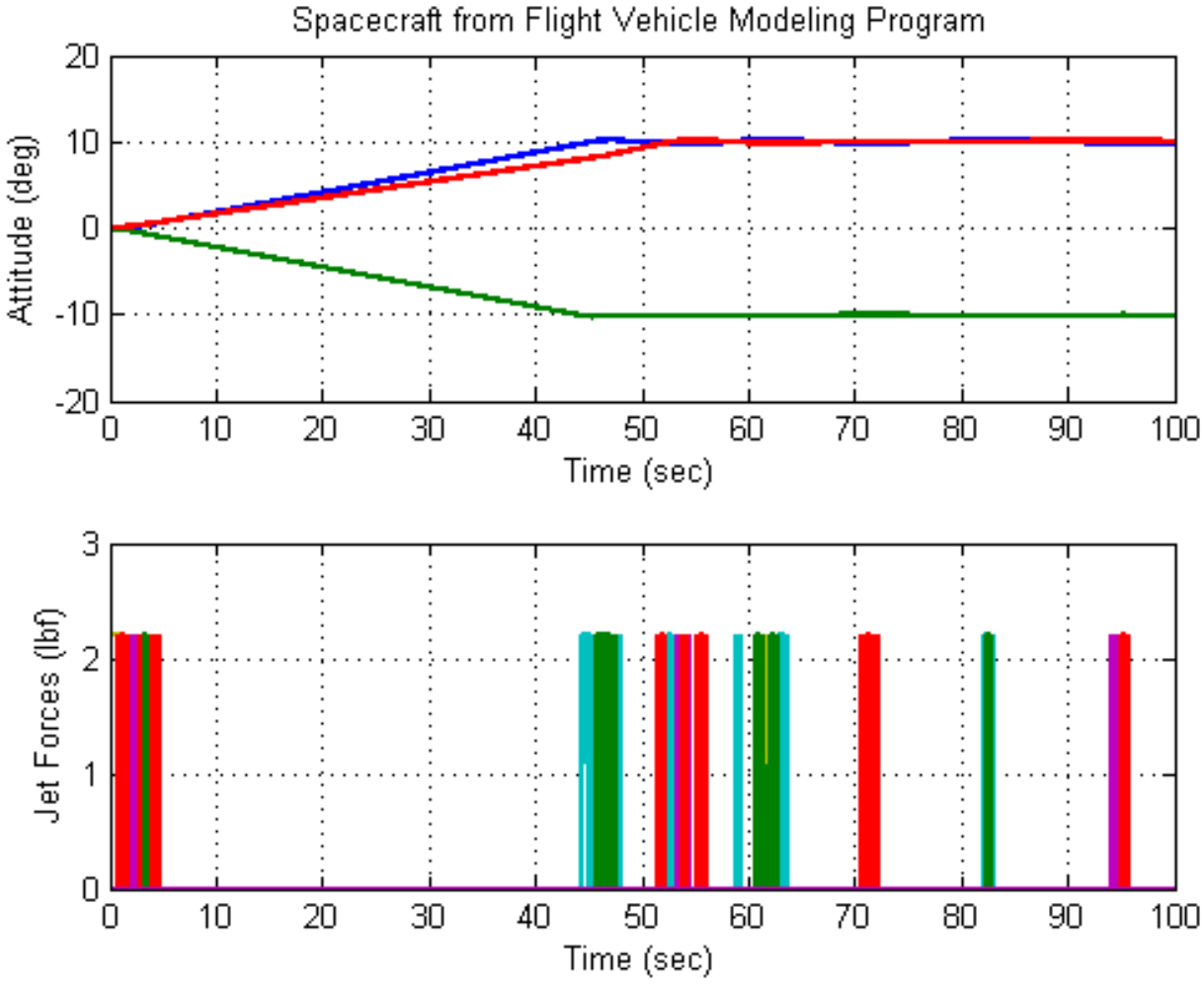


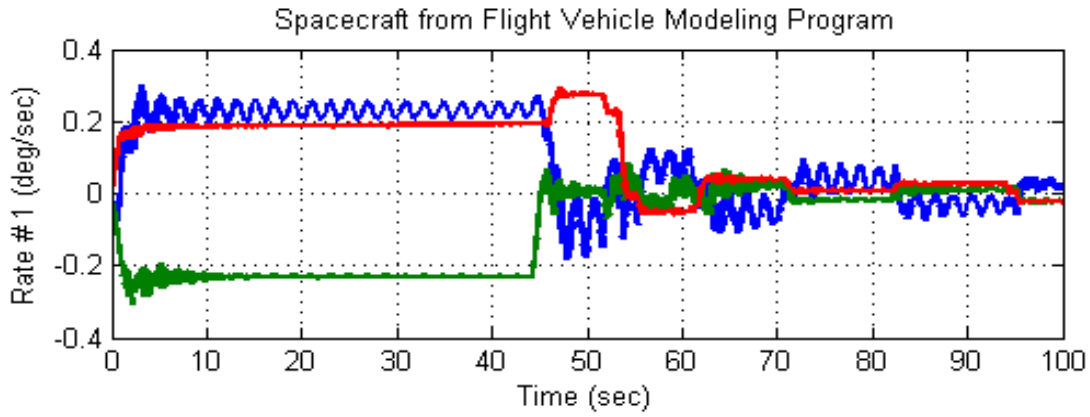

```
% Inputs = 13
%   1   BR-Y+Z dTh/Th for RCS Jet No  1 (-)
%   2   BL+Y+Z dTh/Th for RCS Jet No  2 (-)
%   3   FR-Y-Z dTh/Th for RCS Jet No  3 (-)
%   4   FL+Y-Z dTh/Th for RCS Jet No  4 (-)
%   5   FR-Y+Z dTh/Th for RCS Jet No  5 (-)
%   6   FL+Y+Z dTh/Th for RCS Jet No  6 (-)
%   7   FRAxi-X dTh/Th for RCS Jet No  7 (-)
%   8   FLAxi-X dTh/Th for RCS Jet No  8 (-)
%   9   BR-Y-Z dTh/Th for RCS Jet No  9 (-)
%  10   BL+Y-Z dTh/Th for RCS Jet No 10 (-)
%  11   BRAxi+X dTh/Th for RCS Jet No 11 (-)
%  12   BLAxi+X dTh/Th for RCS Jet No 12 (-)

%  13   Engine Force Direction: (1, 0, 0)
```

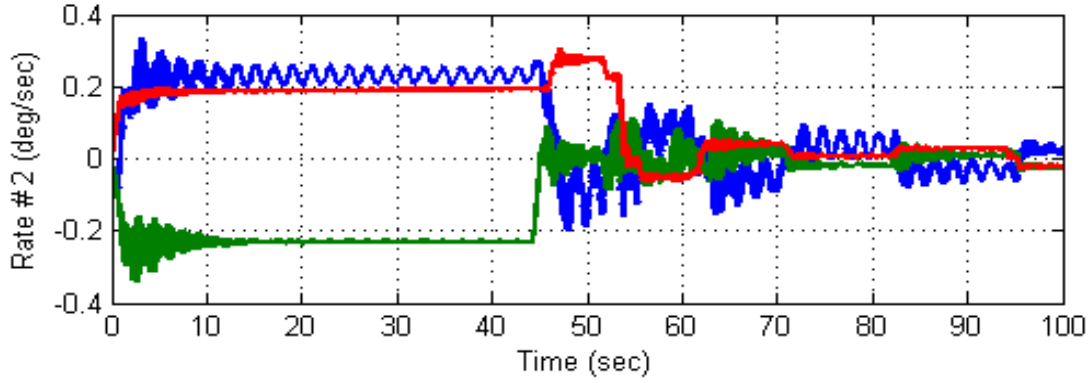
```
% Outputs = 26
%  1   Roll Attitude  (phi-body) (radians)
%  2   Roll Rate      (p-body)   (rad/sec)
%  3   Pitch Attitude (thet-bdy) (radians)
%  4   Pitch Rate     (q-body)   (rad/sec)
%  5   Yaw  Attitude  (psi-body) (radians)
%  6   Yaw  Rate      (r-body)   (rad/sec)
%  7   Angle of attack, alfa, (radians)
%  8   Angle of sideslip, beta, (radian)
%  9   Change in Altitude, delta-h, (feet)
%  10   Forw ard Acceleration  (V-dot)  (ft/sec)
%  11   Cross Range Velocity (Vcr) (ft/sec)

%  12   Gyro #  1, Roll Attitude   (Body) (radians)
%  13   Gyro #  2, Pitch Attitude  (Body) (radians)
%  14   Gyro #  3, Yaw  Attitude   (Body) (radians)

%  15   Rate-Gyro #  4, Roll Rate  (Body) (rad/sec)
%  16   Rate-Gyro #  5, Pitch Rate (Body) (rad/sec)
%  17   Rate-Gyro #  6, Yaw  Rate  (Body) (rad/sec)

%  18   Rate-Gyro #  7, Roll Rate  (Body) (rad/sec)
%  19   Rate-Gyro #  8, Pitch Rate (Body) (rad/sec)
%  20   Rate-Gyro #  9, Yaw  Rate  (Body) (rad/sec)

%  21   Accelerom #  1, (along X), (ft/sec^2) Translat. Acceler
%  22   Accelerom #  2, (along Y), (ft/sec^2) Translat. Acceler
%  23   Accelerom #  3, (along Z), (ft/sec^2) Translat. Acceler

%  24   Accelerom #  4, (along X), (ft/sec^2) Translat. Acceler
%  25   Accelerom #  5, (along Y), (ft/sec^2) Translat. Acceler
%  26   Accelerom #  6, (along Z), (ft/sec^2) Translat. Acceler
```

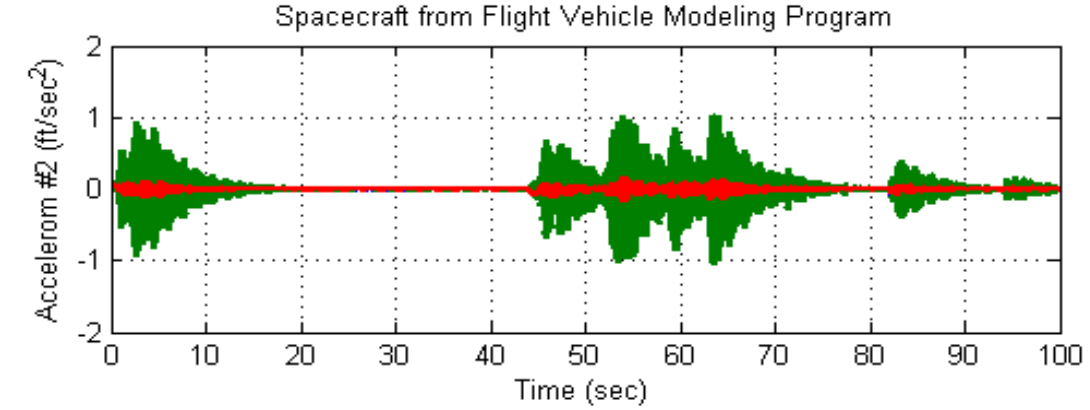
The file "*start.m*" initializes the spacecraft parameters, such as, the jet locations, thrusts, and thrust directions, the phase-plane parameters, the spacecraft CG location, and it also loads the file name "*flex_spacecraft_fvp_s.m*" that contains the flexible spacecraft (a, b, c, d) matrices. This file was created earlier by running the batch. The spacecraft dynamics has 12 jet input forces and one main engine input thrust. Only the jets are used for controlling the spacecraft attitude. The reboost engine force is a disturbance which is counteracted by the RCS jets. The file "*Phase-Plane.m*" contains the phase-plane and jet-selection logic. The file "*pl.m*" is used to plot the data after executing the Simulink model.

Figure (1.2.1) shows the spacecraft rate-gyro, accelerometer, and attitude response to a 10 (deg) attitude command in all three directions. The engine thrust was set to zero. It shows also the rate-gyro and accelerometer responses in two separate locations.




Spacecraft from Flight Vehicle Modeling Program
Rate # 1 (deg/sec)
Time (sec)


Rate # 2 (deg/sec)
Time (sec)


Spacecraft from Flight Vehicle Modeling Program
Accelerom #2 (ft/sec2)
Time (sec)

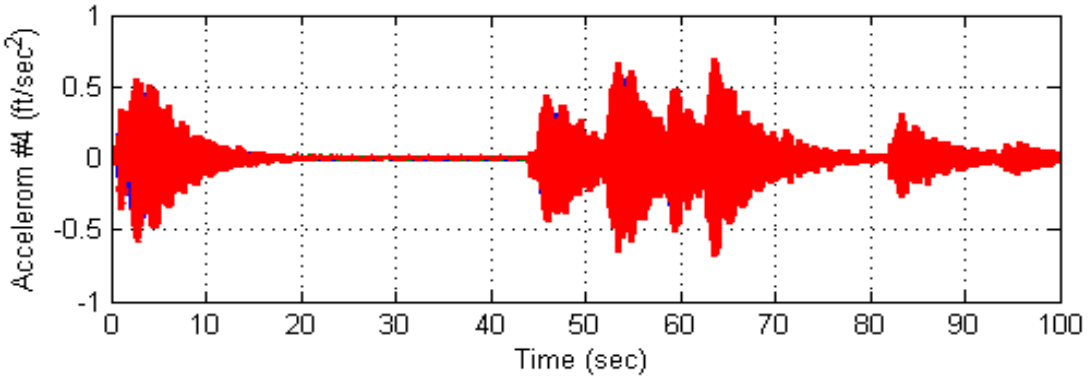

Accelerom #4 (ft/sec2)
Time (sec)

Figure (1.2.2) shows the spacecraft response to the continuous firing of the main engine. The spacecraft is commanded to maintain a constant zero attitude during reboost. The RCS jets are firing regularly attempting to counteract the disturbance torque created by the engine thrust vector and spacecraft CG misalignment and to maintain zero attitude. The attitude is maintained within the dead-band which is 0.2 (deg). The rate-gyro and accelerometer responses in two separate locations are also shown.

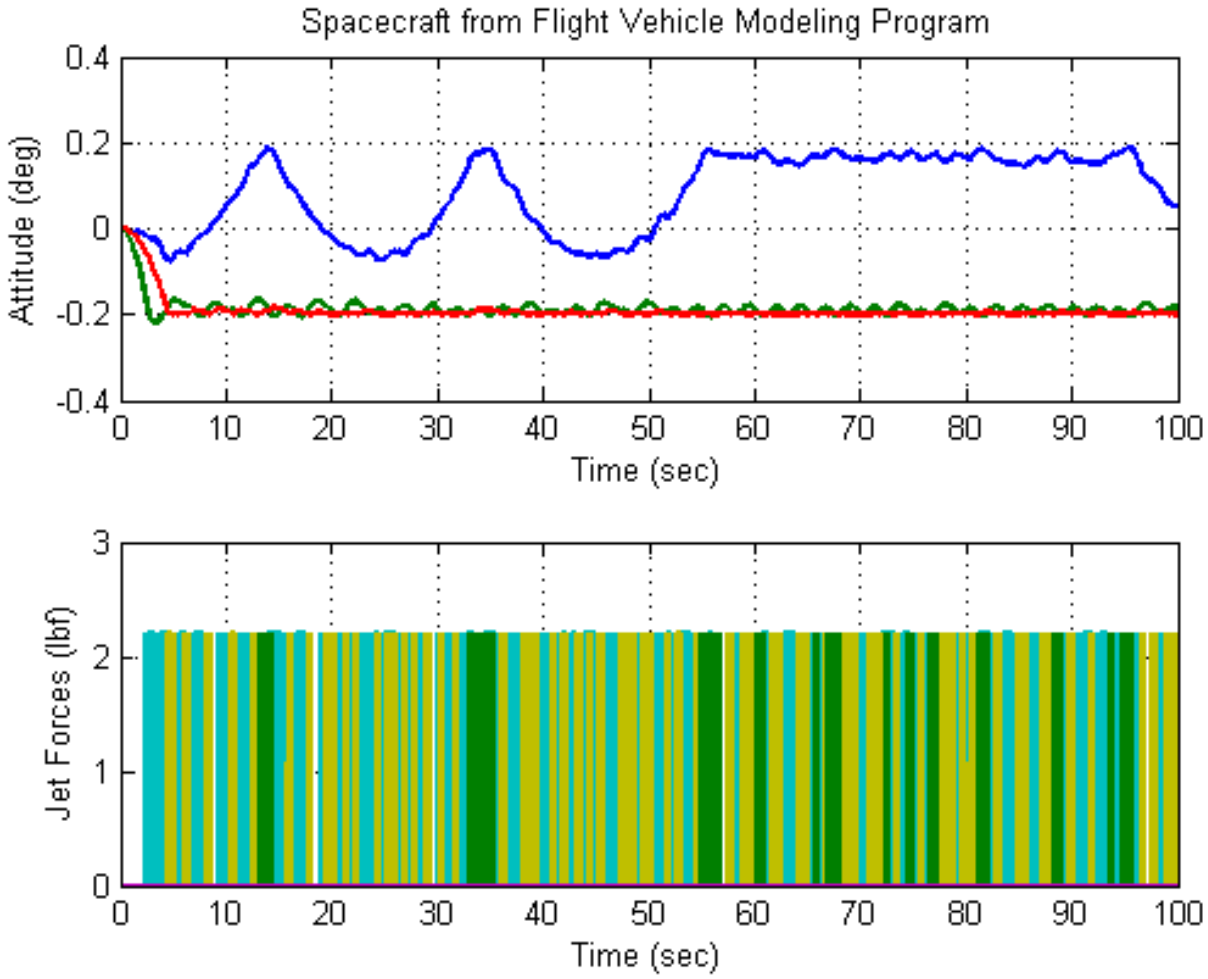
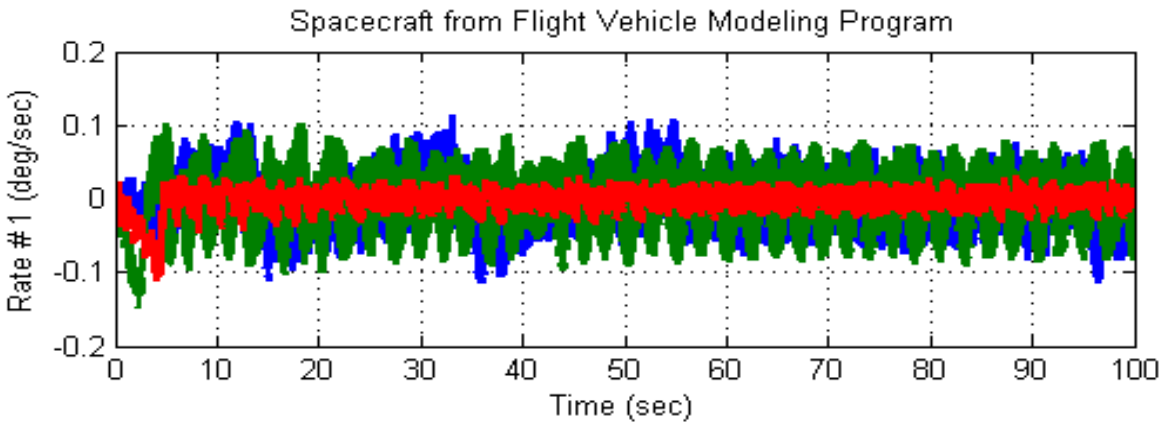
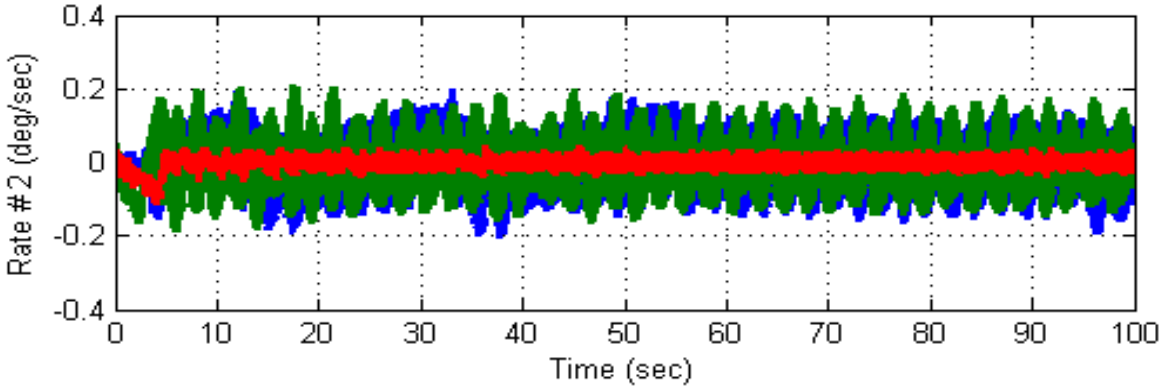
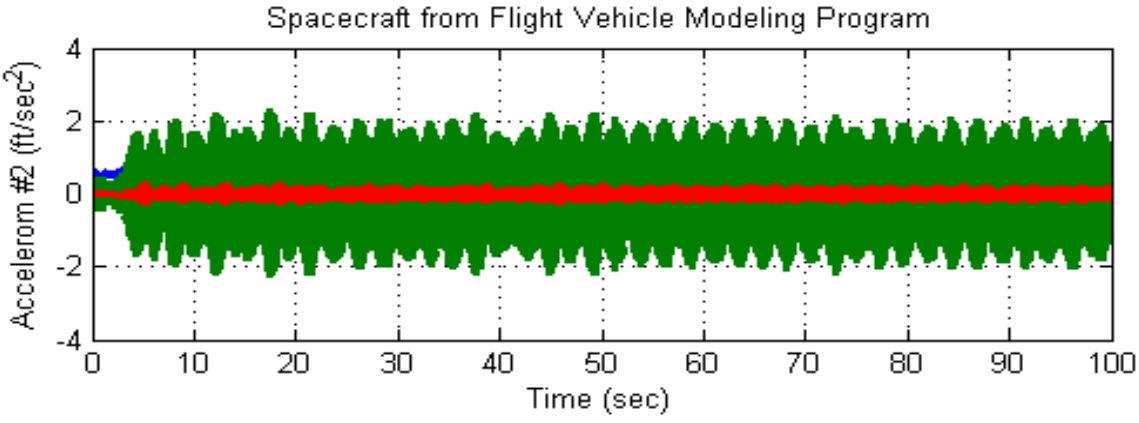
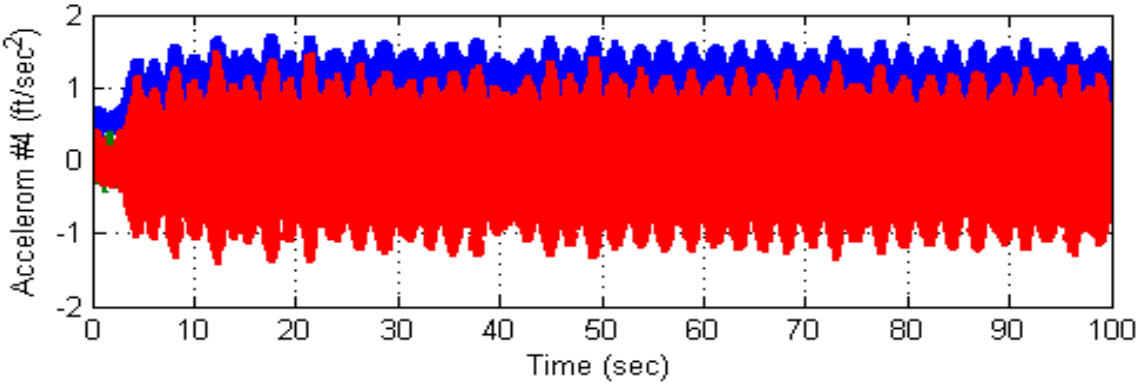


Figure 1.2.2 Spacecraft Attitude response to a continuous firing of the 100 (lb) reboost engine


Spacecraft from Flight Vehicle Modeling Program
Rate # 1 (deg/sec)
Time (sec)


Rate # 2 (deg/sec)
Time (sec)


Spacecraft from Flight Vehicle Modeling Program
Accelerom #2 (ft/sec2)
Time (sec)


Accelerom #4 (ft/sec2)
Time (sec)

2.0 RCS Control

In this section we will design the RCS control laws, starting with a simple phase-plane logic and advancing to a more complex jet selection that minimizes fuel usage. We will also perform analysis and simulations, starting with a simple rigid-body non-linear simulation, and gradually advance to more complex models that include structural flexibility and fuel sloshing. We will present a non-linear model for modeling propellant sloshing at zero or low g, and finally analyze RCS stability in the frequency domain with flexibility and fuel sloshing.

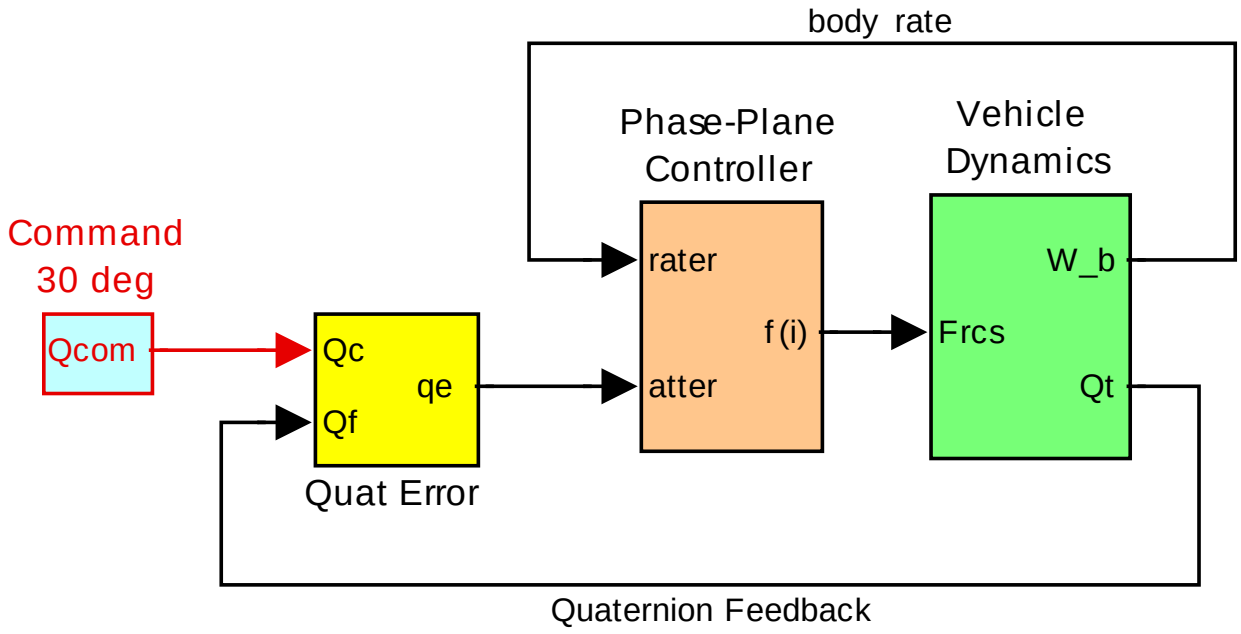


Figure 2.1 Attitude Control System

Let us begin with a simplified version of the RCS Attitude Control System shown in Figure (2.1). It consists of the spacecraft dynamics inside the green block, a phase-plane attitude control system (orange block), a quaternion attitude command generator, and a quaternion error block (yellow).

2.1 Attitude Control System

The orange block in Figure (2.1) is the Attitude Control System (ACS) which consists of the phase-plane logic and the jet-selection logic. The yellow block calculates the attitude error. The inputs to the phase-plane logic are attitude errors, and vehicle rates. The phase-plane calculates the demanded change in vehicle rate, which is a vector about which the vehicle must rotate in order to move from the initial orientation to the commanded attitude. The jet selection logic translates the rate command vector into jet firing. It uses dot product to calculate the torque contributions from all 12 jets in the commanded direction and it selects a few jets that contribute the biggest moment in that direction. The output from jet-select is a vector of 12 jet forces. Most of them are "off". The logic fires between 2 to 4 jets at a time, depending on the commanded direction. Figure (2.2) shows the phase-plane logic in one axis. It determines the acceleration direction from the rate and attitude errors. There are 3 separate phase-planes operating simultaneously for roll, pitch, and yaw axes. Each plane consists of three regions, a region of zero firing, a region of positive jet firing, and a region of negative jet firing in the corresponding direction. The firing decision is made based on the combined rate and attitude error in the direction that reduces error.

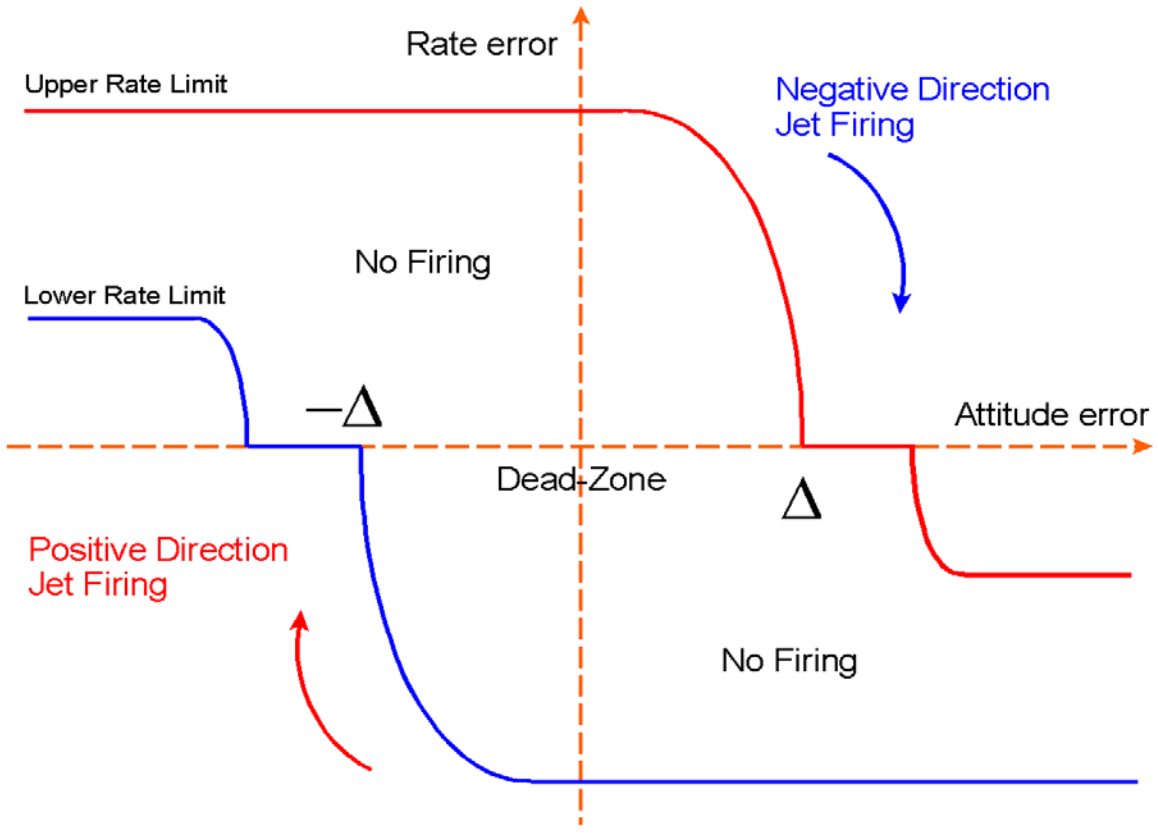


Figure 2.2 Phase-Plane Shows Conditions for Jet Firing

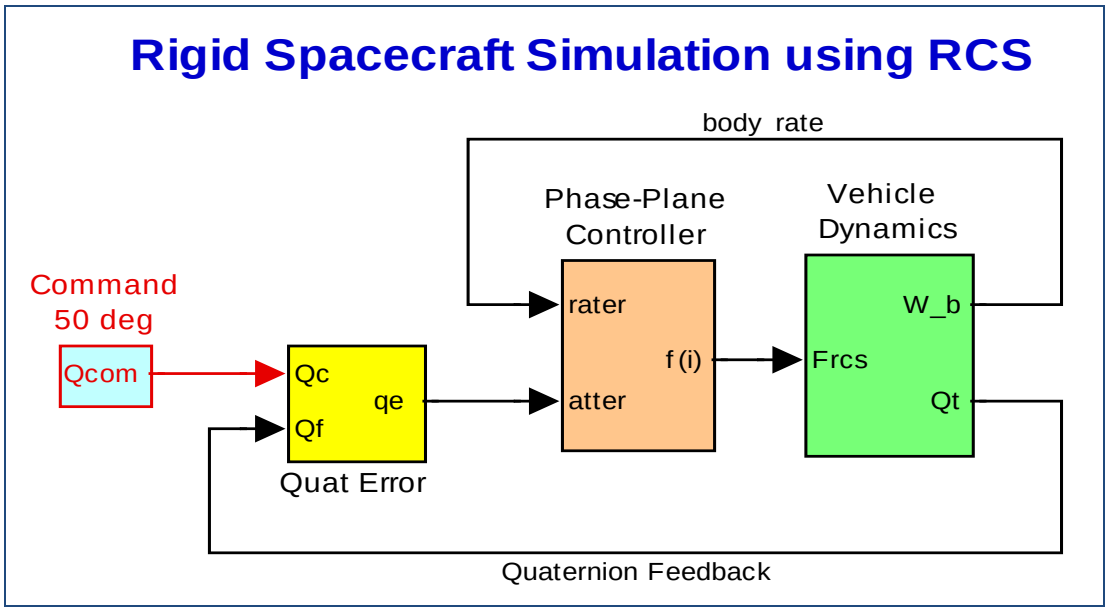
2.2 RCS Simulation Models Using the Dot-Product Jet Selection Logic

In this design example we begin with simple models and gradually upgrade them to more complex ones. As the design progresses we are going to analyze various simulation models with increasing complexity in terms of spacecraft dynamics and also in control design. To avoid confusion, therefore, each simulation model with the associated Simulink and data files are placed in separate folders and they will be analyzed separately. We start with a 3-dof rigid-body ACS simulation that uses dot-product jet selection logic. Then we apply the same ACS control system to the flex models created in Section 1 using two separate Flixan methods. We compare the simulation results obtained from the two models, both in time and also in frequency domain.

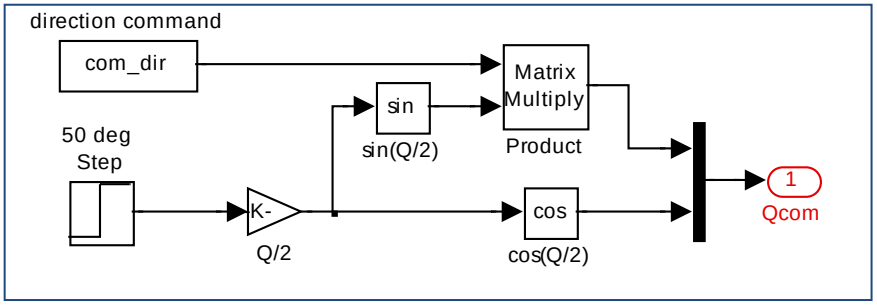
The next step is to upgrade the ACS design and to replace it with a fuel optimal jet selection logic that significantly reduces fuel usage. Following that, we further upgrade the minimum fuel control logic to accommodate also translational in addition to rotational control. We finally test the full 6-dof minimum fuel control logic using a non-linear spacecraft model with flexibility, the flex dynamics being connected in parallel with the previously used non-linear rigid-body model. This demonstrates a wide variety of options which are available for reaction control modeling and analysis.

2.2.1 Rigid-Body Non-Linear Simulation

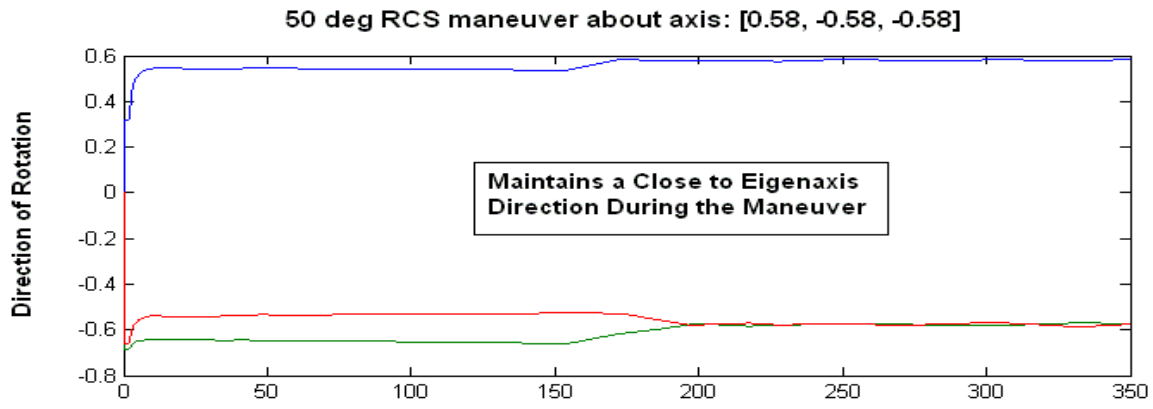
We start the RCS analysis with a simple rigid-body non-linear simulation model in order to demonstrate the phase-plane and jet-selection logic. The files used in this model are in folder "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis**__*(c) NonLin RigBody RCS Attitude Control*__". The Simulink model is "*RB_Sim_RCS.Mdl*" shown in Figure (2.2.1).

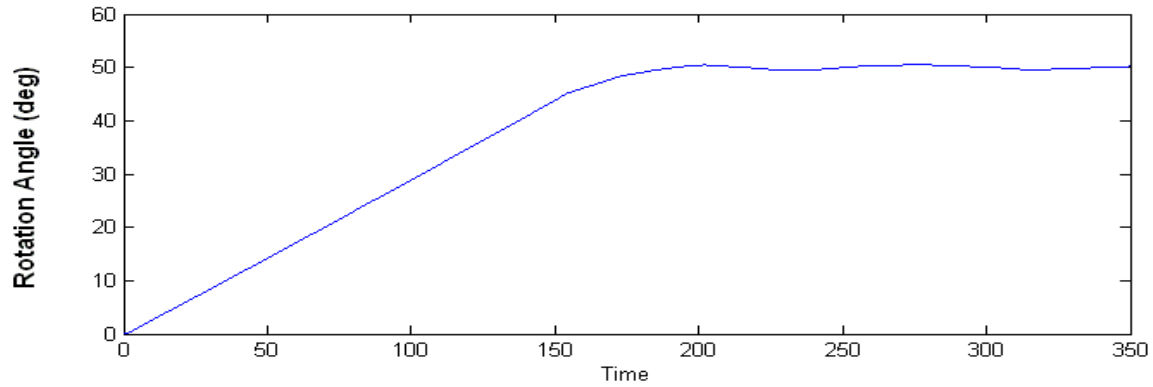


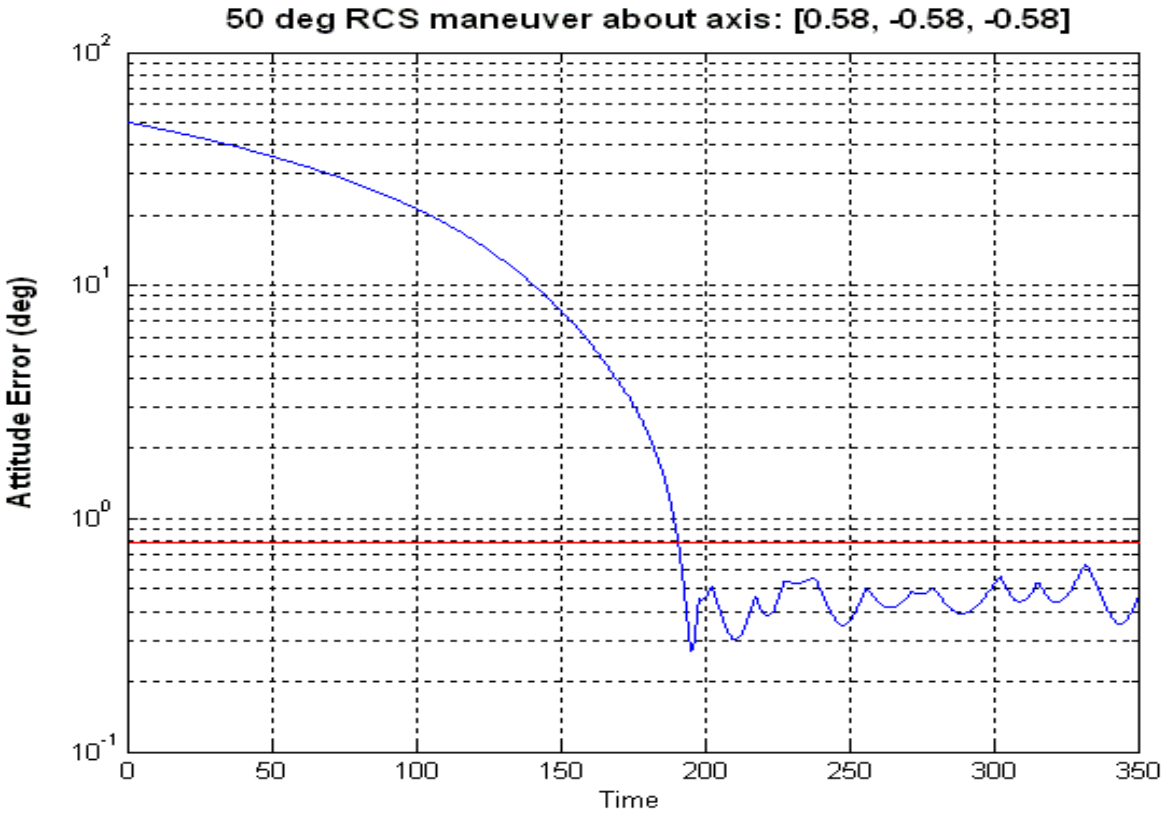
The spacecraft dynamics are implemented in Matlab function "*SV_Dynamics.m*". The spacecraft outputs are body rate and attitude quaternion. The quaternion error (yellow) block receives the attitude quaternion command and the quaternion feedback from the spacecraft and calculates the attitude error which consists only of the 3-axis vector part of the quaternion error, representing attitude errors in roll, pitch and yaw. The magnitude part of the 4-dimensional quaternion error is ignored. The Matlab function "*qerror2.m*" calculates the quaternion error. The phase-plane and jet-selection logic are coded in Matlab functions "*Phase_Plane.m*", and "*Jet_Select_dot.m*". Jet-select is called by the phase-plane logic. The phase-plane parameters such as the dead-band and rate limits are loaded into Matlab workspace by the initialization file "start.m" which must be executed prior to the simulation. Other parameters are also loaded, such as, number of jets, thrust, jet locations, thrust directions, moments of inertia, and cg location. They are used by the jet selection logic to determine which jets should be fired in order to provide rotation in the direction commanded by the phase-plane logic.



The attitude command is defined as a quaternion rotation from the current attitude, which is assumed to be zero, i.e. (0,0,0,1). The quaternion command is a 4-dimentional vector consisting of a direction about which the vehicle should rotate, and the angle of rotation. The block [Qcom] is shown in detail in Figure (2.2.2). The direction "com_dir" is defined in file "start.m", and the rotation angle (50 deg) is defined inside the block. After running the Simulink model, execute file "pl.m" which will plot the simulation results, as shown in figure (2.2.3) below.


50 deg RCS maneuver about axis: [0.58, -0.58, -0.58]
Direction of Rotation
Maintains a Close to Eigenaxis
Direction During the Maneuver
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
50
100
150
200
250
300
350


Rotation Angle (deg)
Time
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350


50 deg RCS maneuver about axis: [0.58, -0.58, -0.58]
Attitude Error (deg)
Time
10^2
10^1
10^0
10^{-1}
0
50
100
150
200
250
300
350

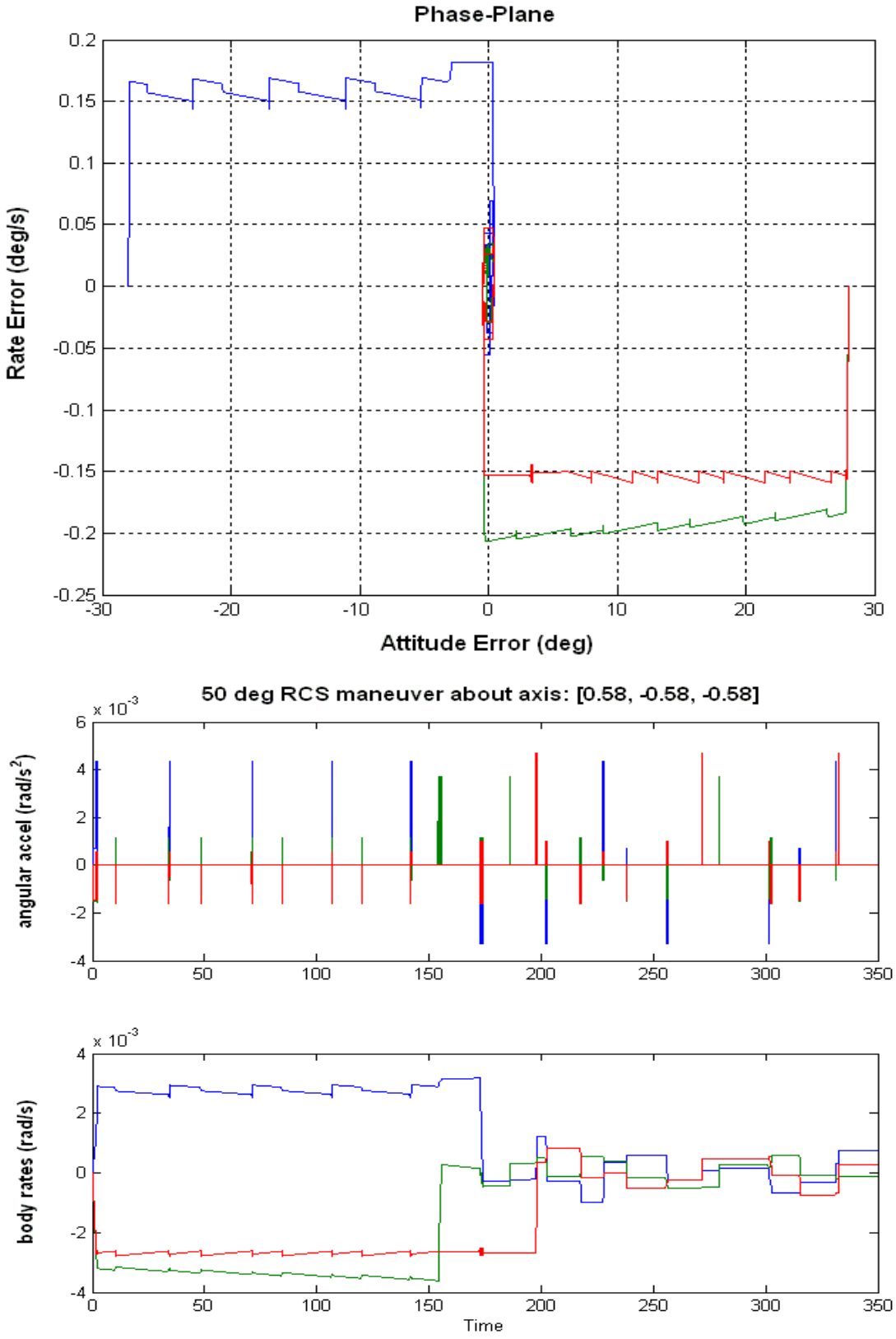


Figure 2.2.3 Rigid-Body Non-Linear System Response to a 50 (deg) command

2.2.2 Comparing the Linear Models with Structural Flexibility in Closed-Loop Sims

The next step in our RCS analysis is to check out by means of simulations the two types of flex spacecraft models that were created in sections 1.1 and 1.2. The first one was created by the flex spacecraft FE modeling program, and the second one was created by the flight vehicle modeling program. We want to make sure that similar results are obtained from both models before we continue any further. The first one has the rigid-body dynamics represented by rigid-body modes from the FEM, while the second one is using its own rigid-body model and only the flex modes are taken out of the FEM. The files for the Simulink models are in the same folder "…*Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis* ***(d) RCS Attitude Control w Flex***". It is a good practice to test and compare these two systems before we move on to more complex models.

Let us begin with the first simulation which is in file "*Sim_Flex_fem_z.mdl*" and uses the FEM, see Figure (2.2.4). The spacecraft system is in file "*flex_spacecraft_fem_z.m*". It includes 40 structural modes and 6 rigid-body modes, a total of 46 modes. It was discretized with a sampling period of T_s=5 msec, and its title is "*RB+Flex Spacecraft with RCS and CMG (Z-Transf)*".

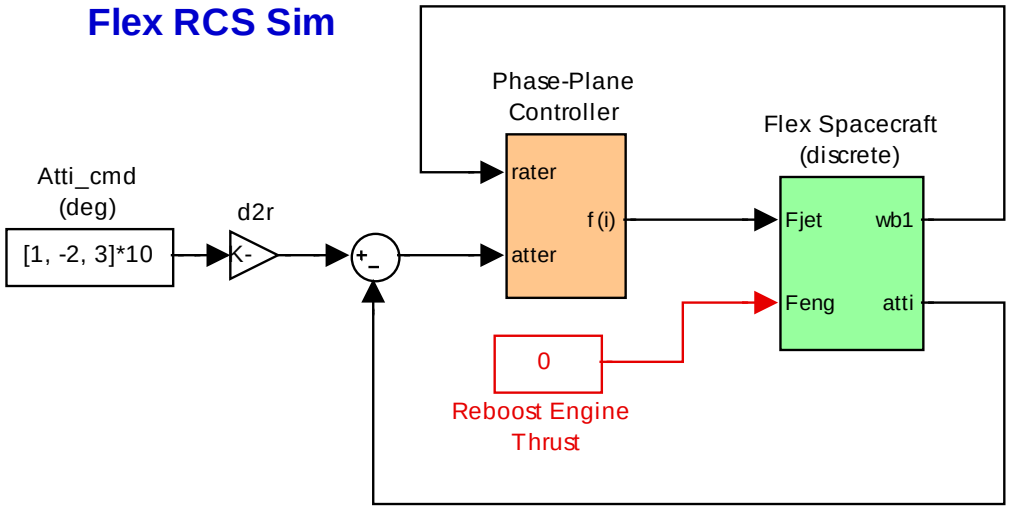


Figure 2.2.4 Linear Flex Spacecraft Simulation "Sim_Flex_fem_z.mdl"

The ACS consists of the phase-plane and the dot-product jet selection logic which is sampled slower, at 20*Ts= 0.1 sec. There is a fuel counter which integrates the sum of jet thrusts: $\int \sum_{i=1,nt} F_{(i)} dt$. The rate transition blocks separate the flex spacecraft (which is sampled at 5 msec) from the ACS (which is sampled every 100 msec). The ACS logic is implemented as a Matlab Function in file "*Phase_Plane.m*", and it is similar to the one described in section 2.2.1. The jet-select output is normalized to unity and it is, therefore, divided by the jet thrust (Th). There is no translation control yet.

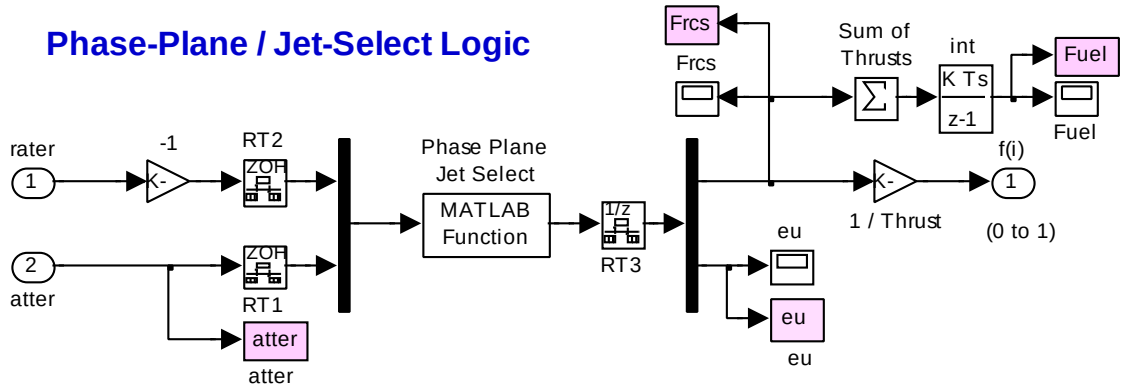


Figure 2.2.5 Phase-Plane and Jet Selection Logic

The discrete flex spacecraft block is shown in detail in Figure (2.2.6). Its inputs are 12 RCS jet forces, and the orbital maneuvering engine thrust (which is not used in this case). The outputs are attitudes, rates, and accelerations at specific vehicle locations defined in section 1.1 during model preparation. File "start.m" initializes the spacecraft and ACS parameters for the simulations, and file "pl.m" plots the simulation data, as shown in figure (2.2.7).

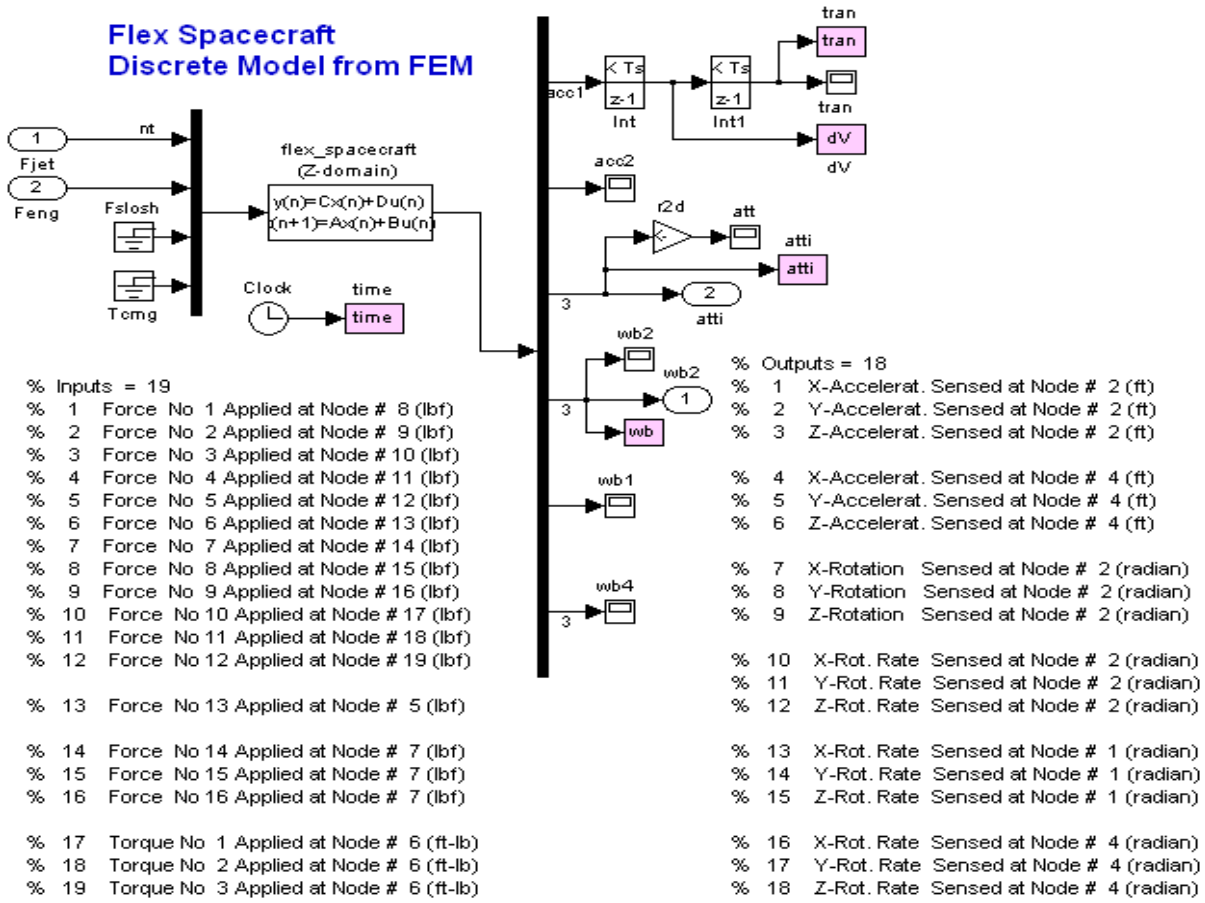
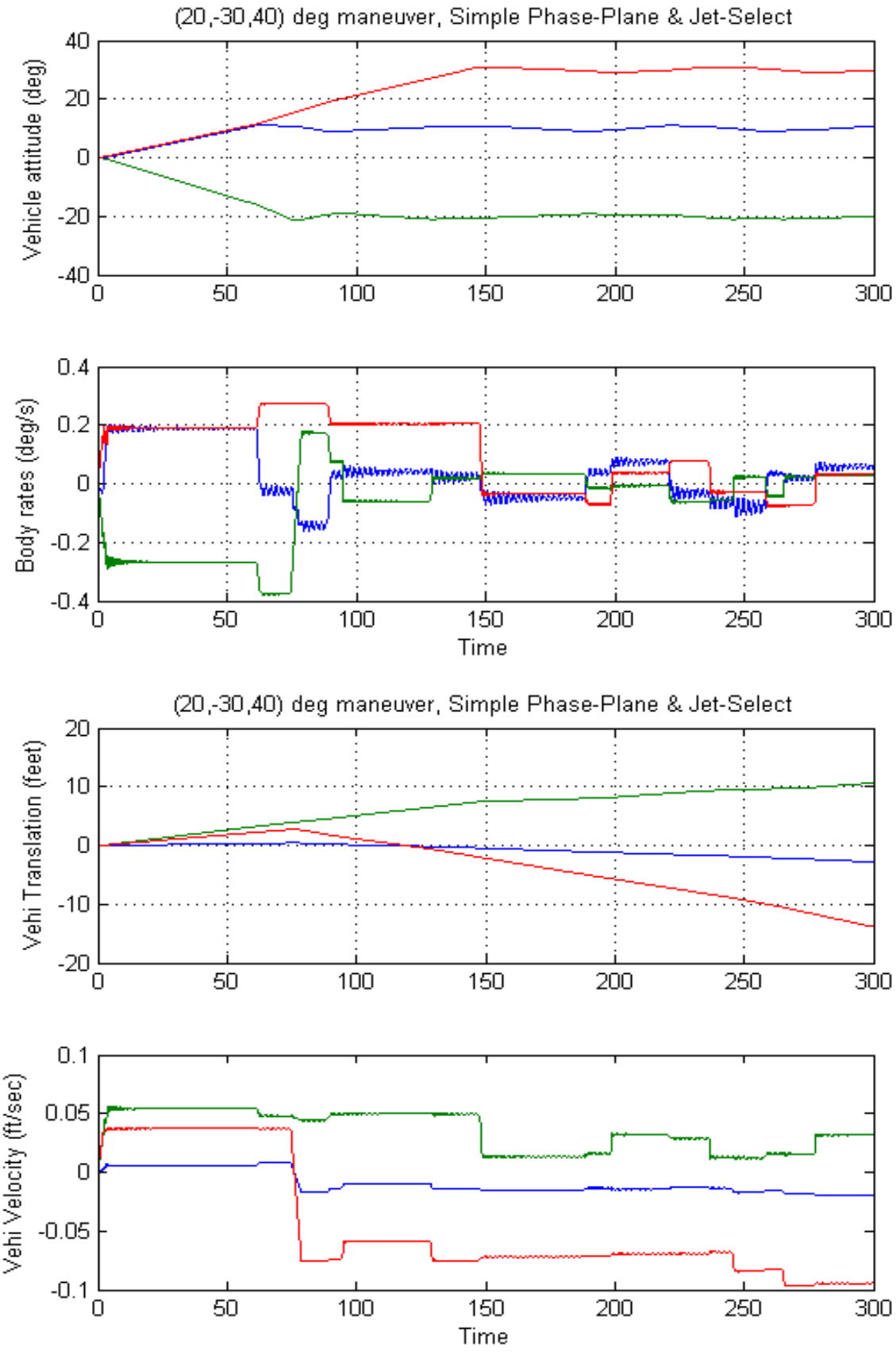


Figure 2.2.6 Discrete Spacecraft State-Space System from file "flex_spacecraft_fem_z.m", system originated from the Flex Spacecraft Modeling Program.


(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehicle attitude (deg)
Body rates (deg/s)
Time
(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehi Translation (feet)
Vehi Velocity (ft/sec)
Time

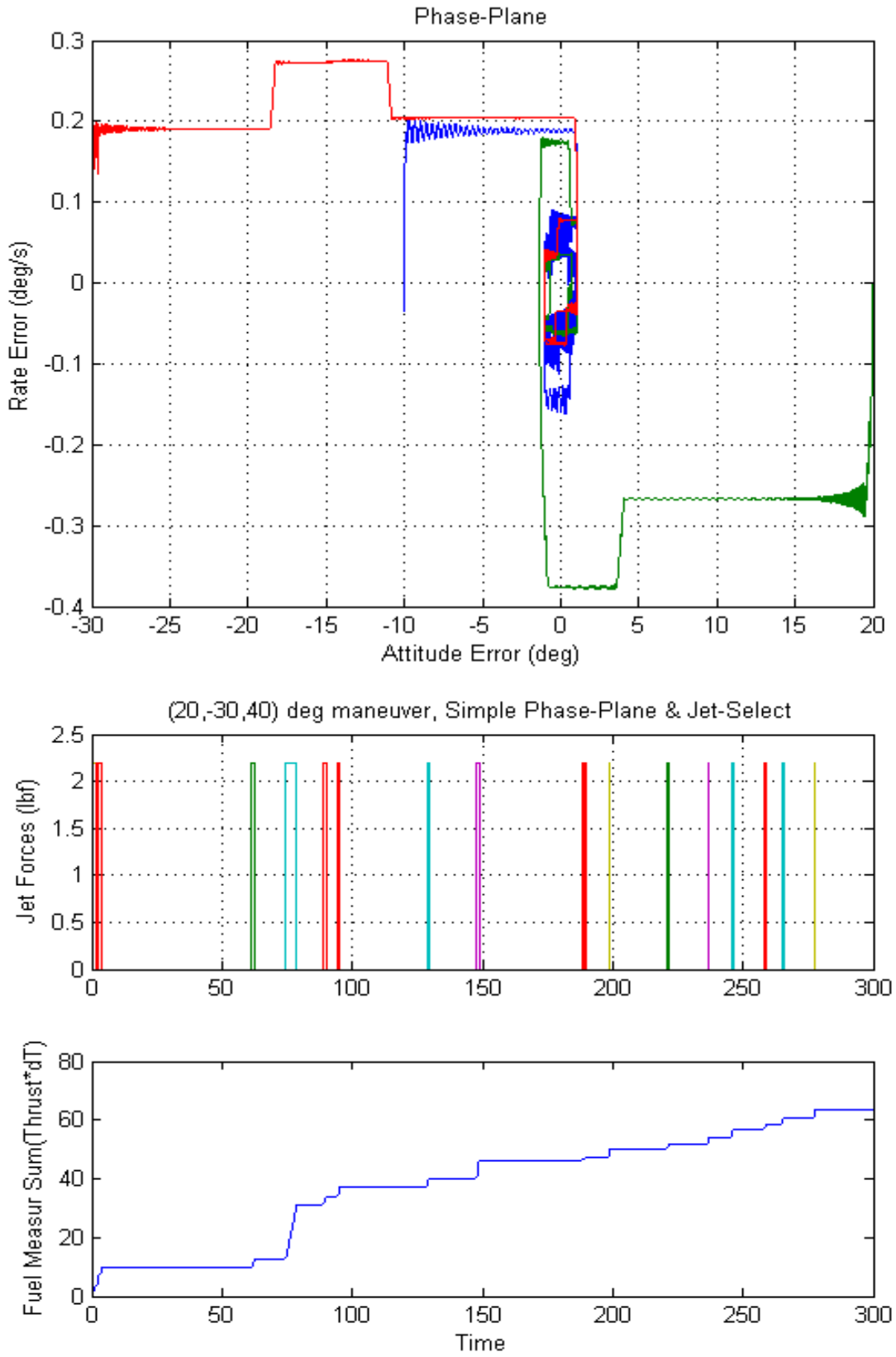


Figure 2.2.7 Simulation Results from "Sim_Flex_fem_z.mdl"

The second Simulink model "*Sim_Flex_fvp_z.mdl*" is almost identical to the first one, but uses the discrete system "*flex_satellite_fvp_z.m*", which was created by the "Flight Vehicle Modeling Program" in Section 1.2. The spacecraft dynamics block is shown in detail in Figure (2.2.7).

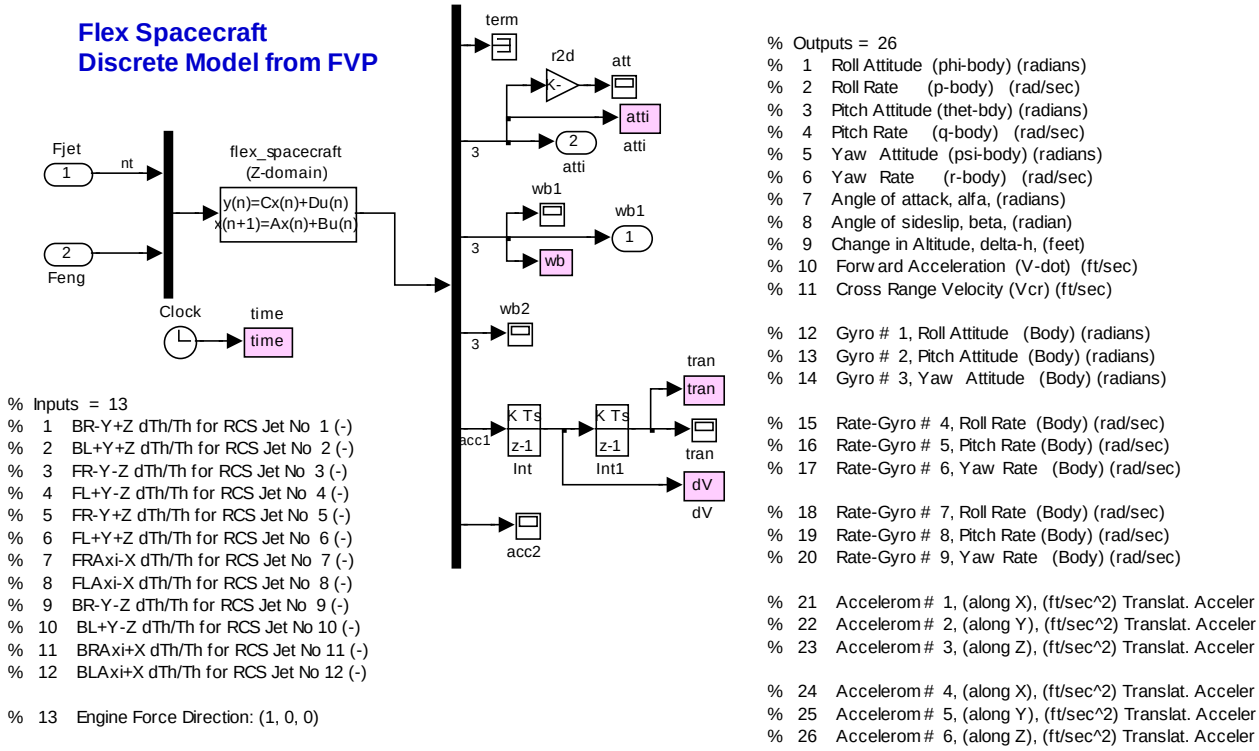
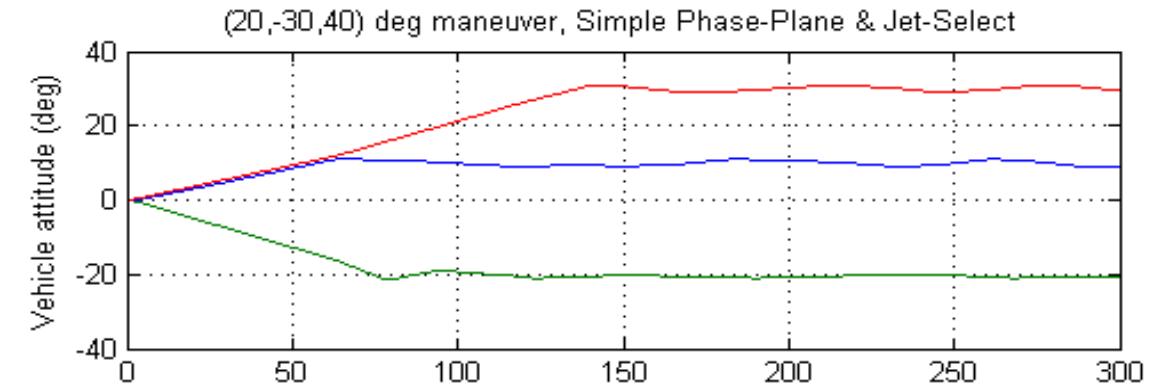
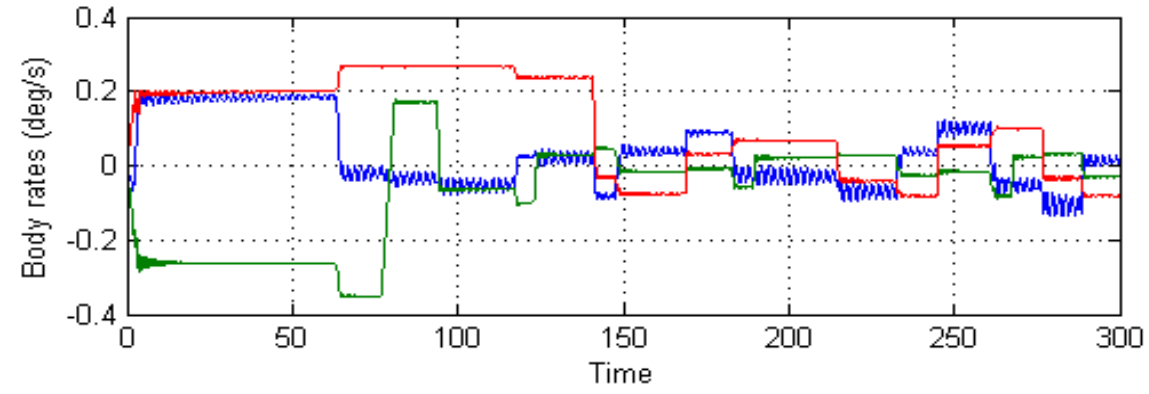


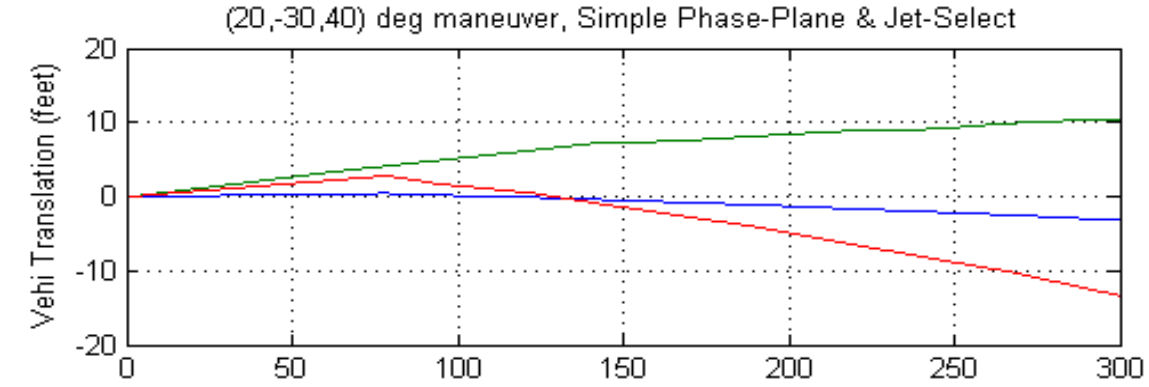
Figure 2.2.7 Discrete Spacecraft State-Space System from file "flex_satellite_fvp_z.m", system originated from the Flight Vehicle Modeling Program.

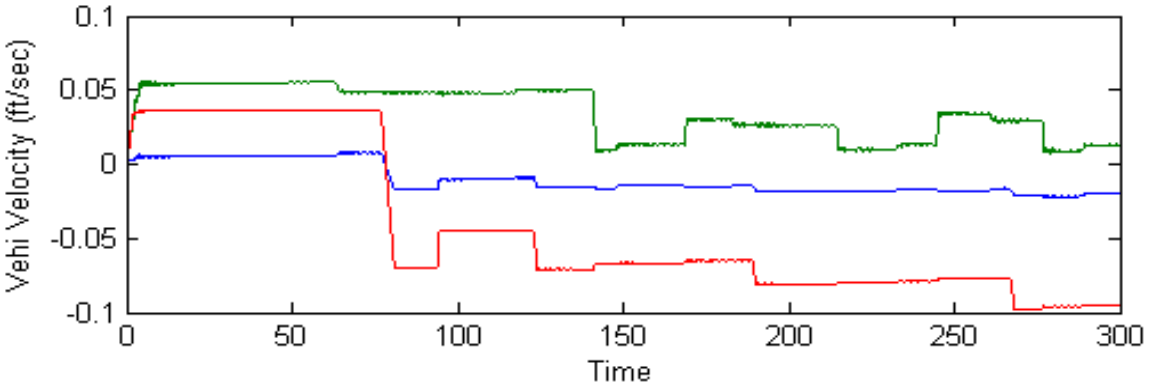
The jet-selection logic output in this model is a little different, because, it uses the FVP spacecraft system. This system requires the jet thrust inputs to vary between zero and one (one representing max thrust). The actual thrust value is integrated in the state-space system data. The file "start.m" initializes both models, and file "pl.m" plots the results from either simulation after completion.

The following plots show the spacecraft response to attitude commands. The attitude converges to its commanded position. The rates are limited to approximately 0.2 (deg/sec). It is significant to notice that at the end of the maneuver the vehicle linear velocity and position are not zero. This will be corrected later by including a translational control logic. The plots show that the responses of the two models to attitude commands are almost identical. There may be negligible differences between the two models in the rigid-body data.


(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehicle attitude (deg)
40
20
0
-20
-40
0
50
100
150
200
250
300


Body rates (deg/s)
0.4
0.2
0
-0.2
-0.4
0
50
100
150
200
250
300
Time


(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehi Translation (feet)
20
10
0
-10
-20
0
50
100
150
200
250
300


Vehi Velocity (ft/sec)
0.1
0.05
0
-0.05
-0.1
0
50
100
150
200
250
300
Time

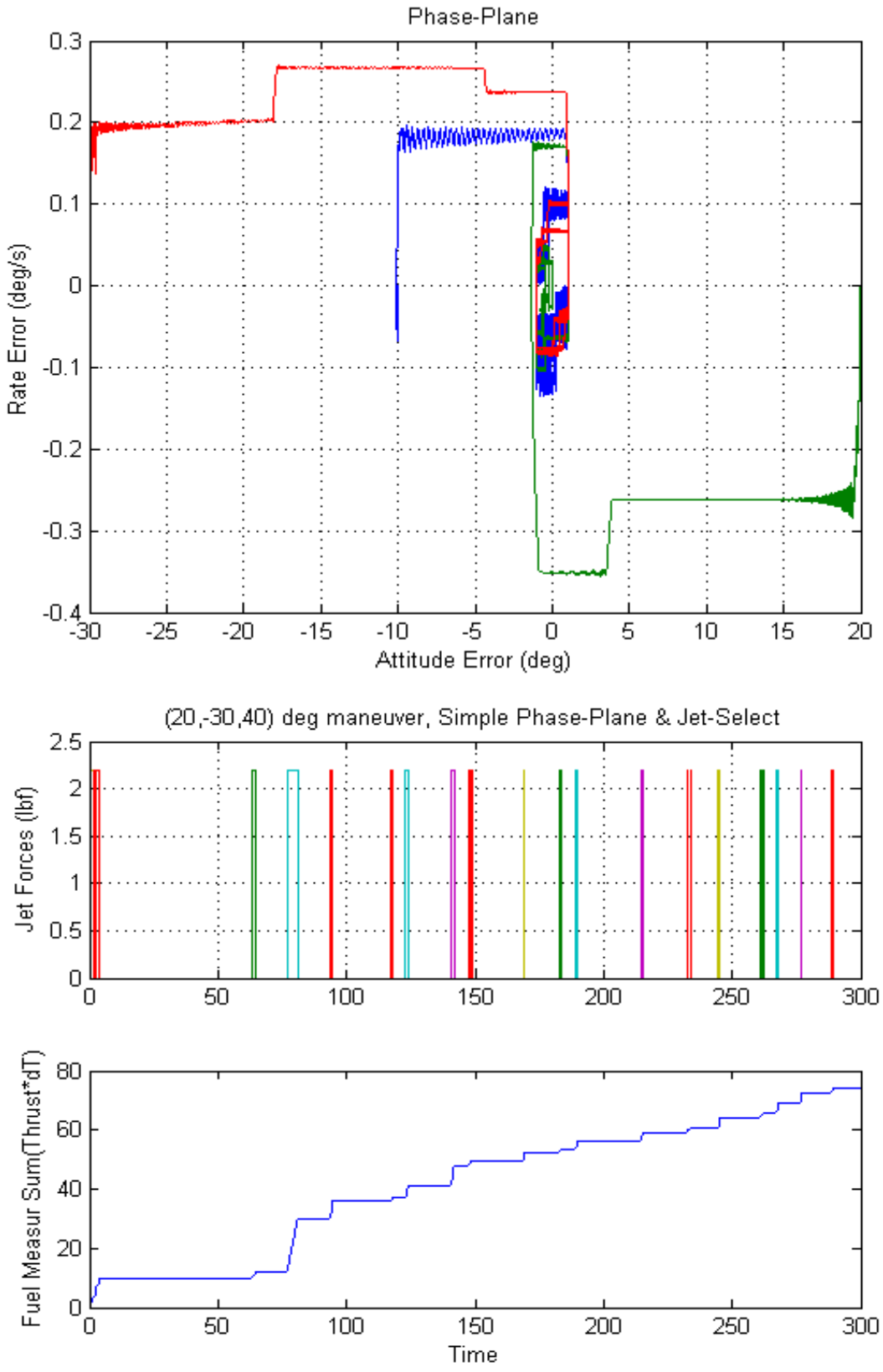
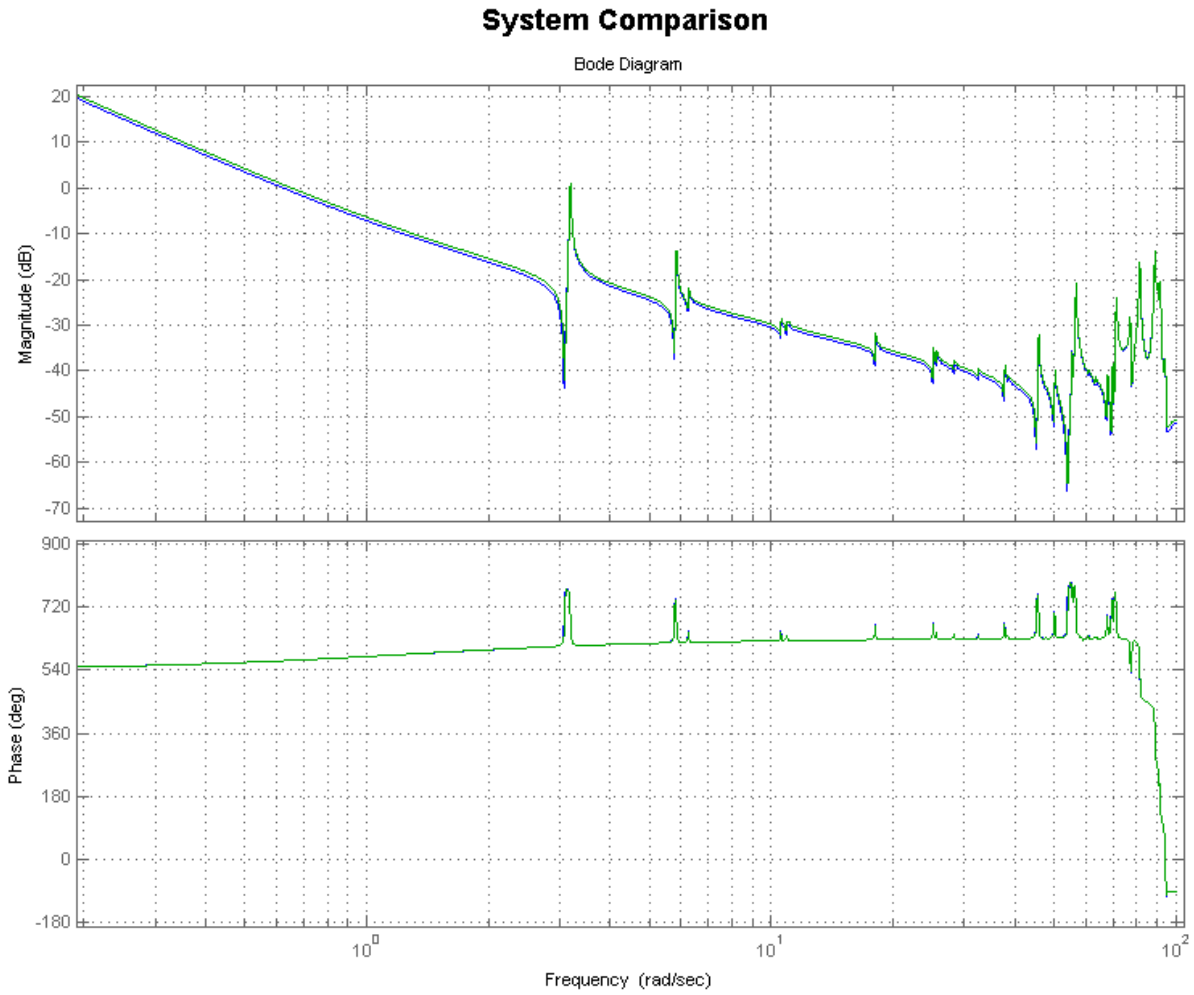


Figure 2.2.7 Simulation Results from "Sim_Flex_fvp_z.mdl"

System Comparison in the Frequency Domain

Move down to the subfolder "*Frequency Domain Comparison*" and run the Matlab file "*run_frequ.m*". This file loads the two spacecraft systems "*flex_spacecraft_fem_s.m*" and "*flex_spacecraft_fvp_s.m*" that were created using different methods and calculates the frequency responses of the open-loop systems including linearized controls. The frequency responses are shown plotted together in Bode and Nichols charts in Figure (2.2.8). The results are almost identical proving a very good match between the two modeling approaches. This is an encouragement to continue the analysis further.



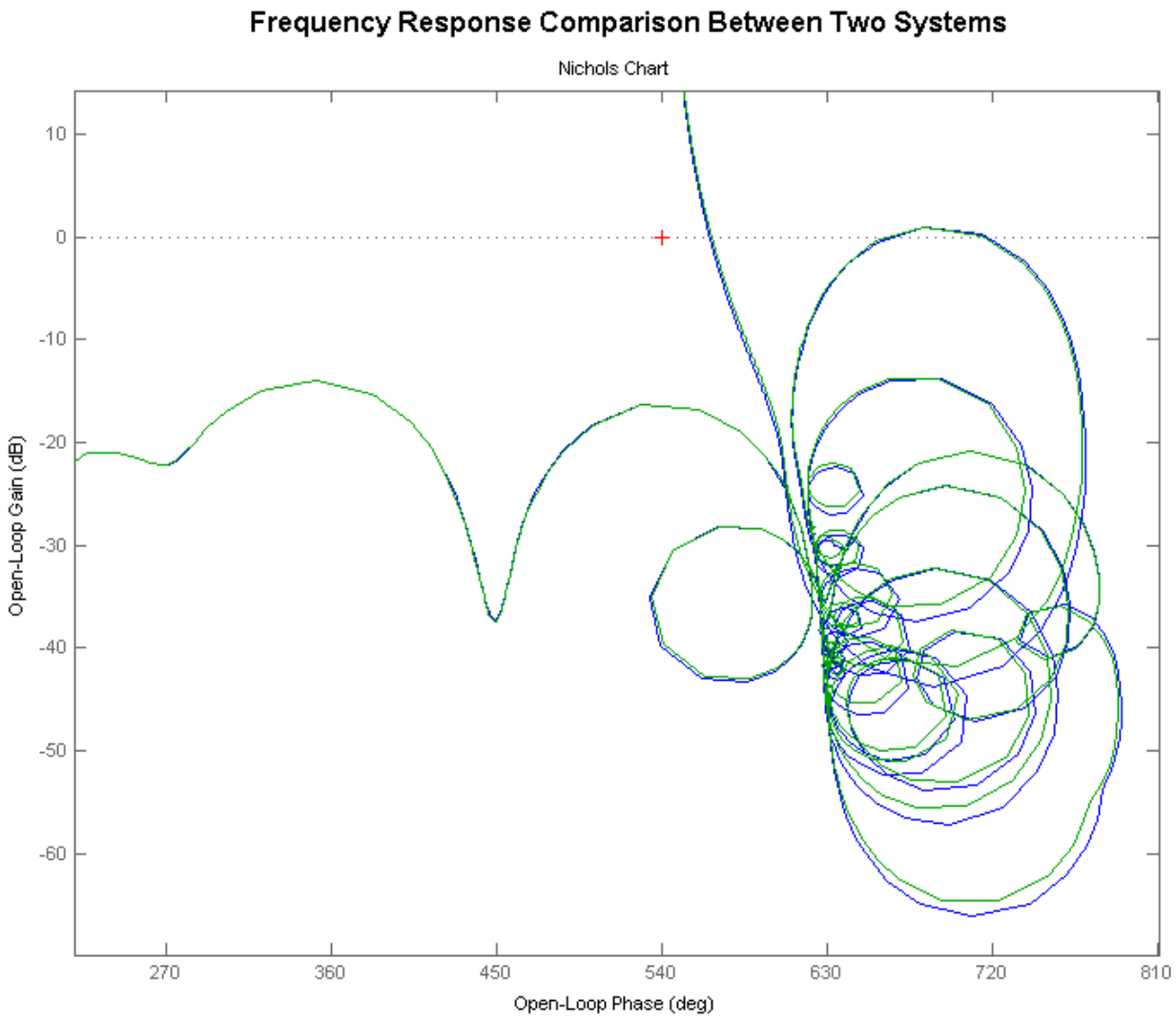


Figure 2.2.8 Frequency domain plots show a very good comparison between the two models

2.4 Jet Selection Logic that Minimizes Fuel Usage

A typical attitude control Reaction Control System (RCS) consists of the phase-plane logic and the jet selection logic. The phase-plane logic determines the direction where to apply a torque on the spacecraft in order to correct for attitude errors and it generates a rate change command to the jet selection logic. In a 12 jet, dot-product based, jet selection logic, 2 to 4 jets are selected which are contributing a positive torque in the demanded direction, and the logic fires those jets for a short period to correct the attitude error. The direction of motion, however, may be near but it is rarely along the commanded direction, which causes an error to develop in a different direction which corresponds to a new set of jets to be selected and fired in the next cycle. It is common sense, therefore, to assume that in order to rotate the spacecraft about a commanded direction more optimally in terms of fuel usage, after selecting the jets which contribute a positive torque in that direction, not all of them should be fired at the same level. Instead, if we assume that the flow rate is proportional to thrust, the ones which contribute more torque should be allowed to fire at higher thrusts than the ones which contribute less in that direction. But this is impossible because most reaction control jets can only fire at constant thrust. They are either "on" or they are "off." In this case we can use pulse width modulation. We assume that the control cycle period is relatively long enough (0.1 sec) to allow sufficient room for a jet pulse width modulation within the cycle. In the beginning of the control cycle we select 3 jets and turn them on together. Then we allow the most contributing jets to stay on longer within the cycle than the less contributing jets, an "on-time" proportional to the amount of contribution of the selected jets. The jet control logic commands the jets every 0.1 sec. When a jet is selected it receives also its "on-time", which is in multiples of 5 msec, a minimum of 5 msec, and a maximum of 95 msec. So the attitude control software does not have the responsibility to turn "off" the jets.

But how do we determine how long should each of the selected jets stay "on" during the control cycle? Let us assume that three jets out of 12 were selected (i, j, and k) and their corresponding vehicle acceleration vectors from each jet are: (a_i , a_j , a_k) respectively. If during the cycle we turn them on for a period of (t_i , t_j , t_k) respectively, and there are no other disturbances, at the end of the cycle the vehicle rate ($\delta\underline{\omega}$) will be

$$\delta\underline{\omega} = \begin{pmatrix} \underline{a}_i & \underline{a}_j & \underline{a}_k \end{pmatrix} \begin{bmatrix} t_i \\ t_j \\ t_k \end{bmatrix} = \left[A_{i,j,k}\right]\underline{t} \qquad (2.4.1)$$

Now if ($\delta\underline{\omega}_c$) represents the commanded change in vehicle rate at the end of the cycle, and that the vehicle does move much during the short cycle, then we can invert the matrix and solve for the on-times. We assume of course that the jets were properly selected for the commanded direction to provide positive on-times. Otherwise, it may result into negative on-times.

$$\underline{t} = \left[A^{-1}_{i,j,k}\right]\delta\underline{\omega}_c \qquad (2.4.2)$$

Assuming that (f_i , f_j , f_k) are the corresponding amounts of fuel flow rates for the three thrusters selected, the total amount of propellant used by jets (i, j, and k) to achieve a commanded change in rate ($\delta\underline{\omega}_c$) is

$$p_{i,j,k} = \begin{pmatrix} \underline{f}_i & \underline{f}_j & \underline{f}_k \end{pmatrix} \begin{bmatrix} t_i \\ t_j \\ t_k \end{bmatrix} = \begin{pmatrix} \underline{f}_i & \underline{f}_j & \underline{f}_k \end{pmatrix} \left[A^{-1}_{i,j,k} \right] \delta\underline{\omega}_c \qquad (2.4.3)$$

The propellant usage factor $p_{i,j,k}$ is the criterion used for selecting 3 jets from a total of 12 jets. The thrust directions of the jets on the spacecraft are assumed to be properly selected so that there are always 3 jets, at least, which provide a positive torque in any commanded direction. For a given change in rate command ($\delta\underline{\omega}_c$), the jet-select logic first chooses 3 jets that minimize equation (2.4.3) using function **Jet_Select_3dof**, and it calculates also the "on-times" from equation (2.4.2).

2.4.1 Upgrading the ACS Model with the Min Fuel Logic

The Simulink model "*Sim_Flex_3rot.Mdl*" shown in Figure (2.4.4) implements the minimum fuel attitude control logic. It can be found in folder "*\Flixan\ Examples\Flex Agile Spacecraft with SGCMG & RCS\Analysis* ***(e) Min Fuel RCS Control 3-Rot Flex***". It consists of the RCS attitude control system which operates at 0.1 sec sampling period, and the flexible spacecraft dynamics which is sampled every 5 msec. The roll, pitch, yaw attitude command is applied on the left side.

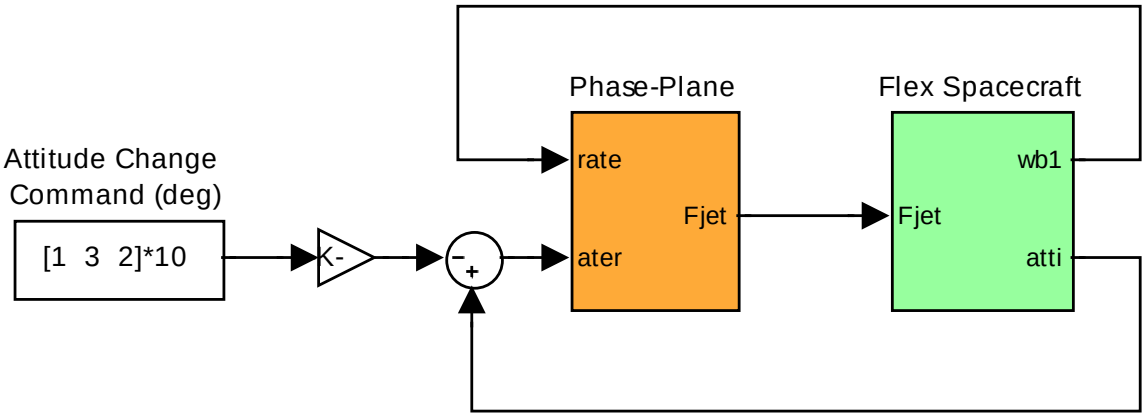


Figure 2.4.4 The 3-dof Simulation Model (Sim_Flex_7R.Mdl) for Min Fuel ACS

The orange ACS block is shown expanded in Figure (2.4.5). It consists of the phase-plane logic which is implemented in Matlab function "*Rotat_PP3.m*". It receives the attitude error and body rate signals. The phase-plane calculates the commanded change ($\delta\underline{\omega}$) in vehicle rate (roll, pitch, and yaw), that feeds into the jet selection logic, shown in Figure (2.4.6), which turns the thrusters "on" or "off" as needed to maneuver the vehicle. Since the system operates at two different rates, 0.005 sec and 0.1 sec, rate transitioning blocks are used in the interfaces between the Simulink blocks.

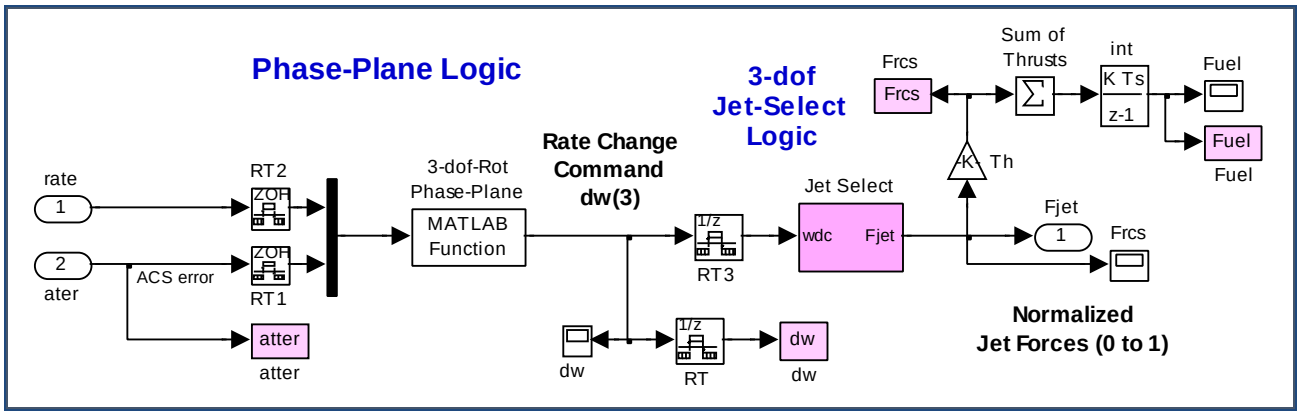


Figure 2.4.5 Attitude Control System consisting of Phase-Plane, and Jet-Select Logic

Figure (2.4.6) shows the implementation of the jet selection logic. The function f= **Jet_main_3dof**(sync, δω), is called by the Simulink model jet selection logic block, and calls the function [js, t]= **Jet_Select_3dof**(δω) which performs a jet search, and selects 3 among the 12 jets that minimize equation (2.4.3). The input (δω) is the change in vehicle rate (in roll, pitch, and yaw) commanded by the phase plane logic. The function outputs are the selected jet numbers the array js(.) and the corresponding on-times for each jet in array t(.).

The input "sync" is a saw-tooth signal that synchronizes the jet turning-off times in the simulation, not in the actual hardware. In the beginning of the control cycle the 3 selected jets are turned "on". They remain on until the logic turns them "off" sequentially, within the 0.1 sec control cycle, according to their "on-times" t(i). The cycle repeats every 0.1 sec interval with new jets and on-times. The turning-off times during the 0.1 sec interval occur in multiples of 5 msec. In the actual hardware implementation each jet that is turned on receives also its "on-time" in the beginning of the cycle, and it has the logic to turn itself "off" when its on-time expires.

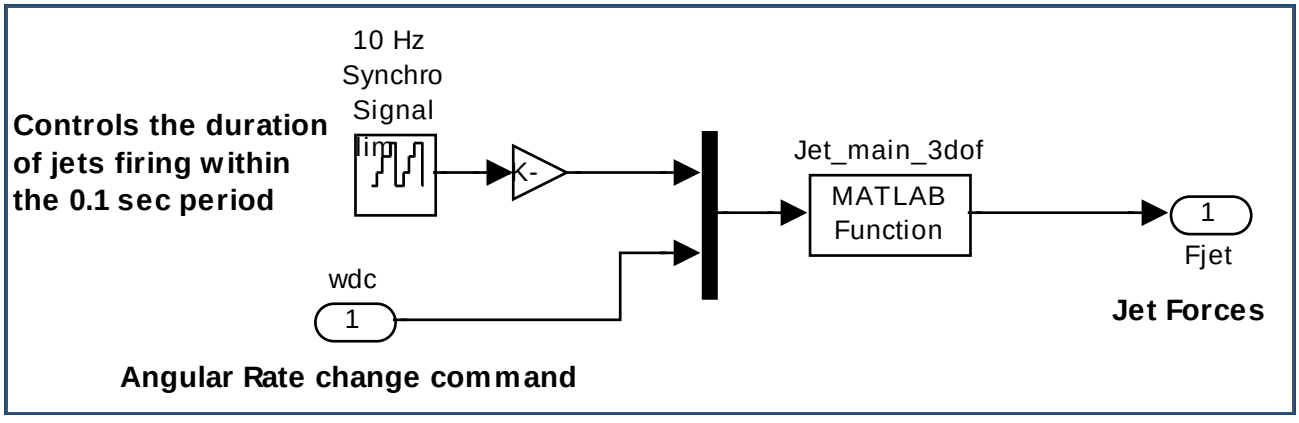


Figure 2.4.6 Minimum Fuel, 3-dof, Jet Selection Logic

The flexible spacecraft model is shown in Figure (2.4.7). It contains the discretized system "*Large Flexible Spacecraft with RCS (Z-Transf)*" described by state-difference matrix equations sampled every 5 msec. It is loaded from file "*flex_spacecraft_fvp_z.m*". The inputs to the system are the 12 jet forces. The orbital maneuvering engine is not used in this case. We will use it later when we analyze fuel sloshing. The outputs are attitude rates, and accelerations at the navigation base plus at other locations already described. The vehicle translation is approximated by integrating the acceleration twice.

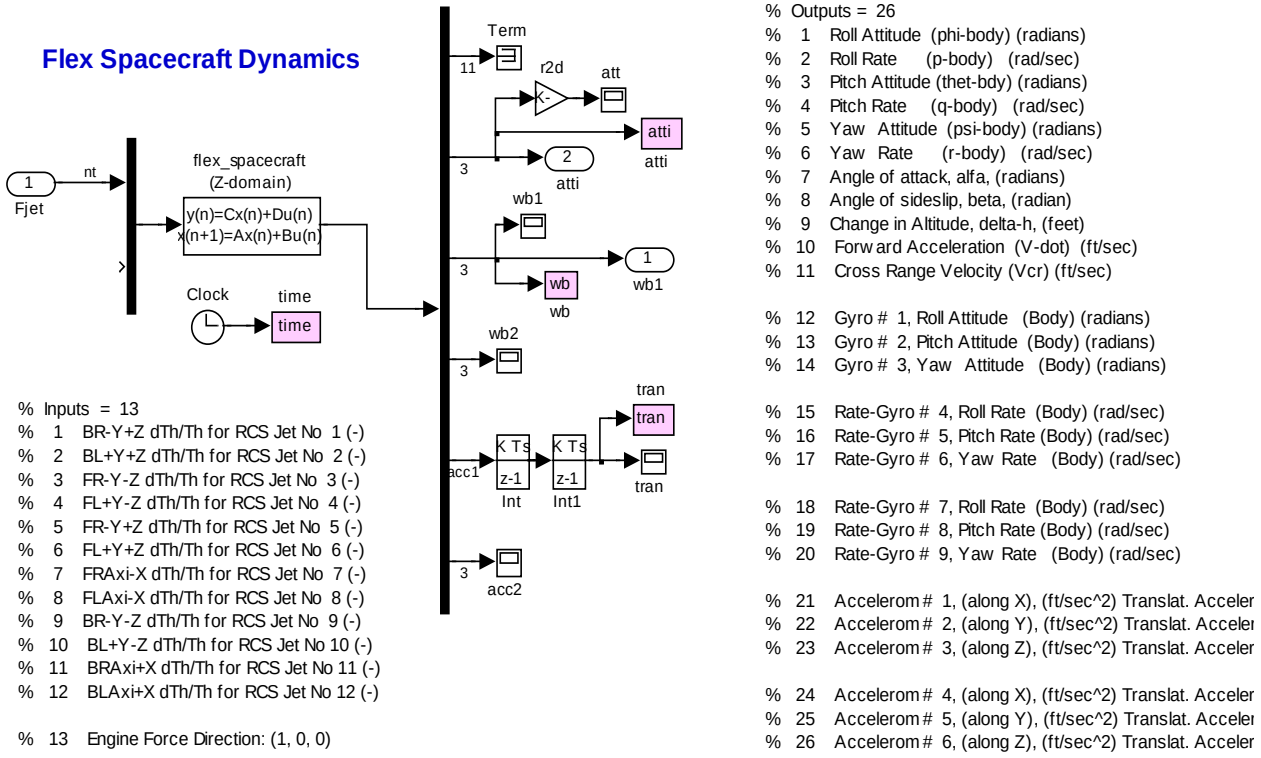
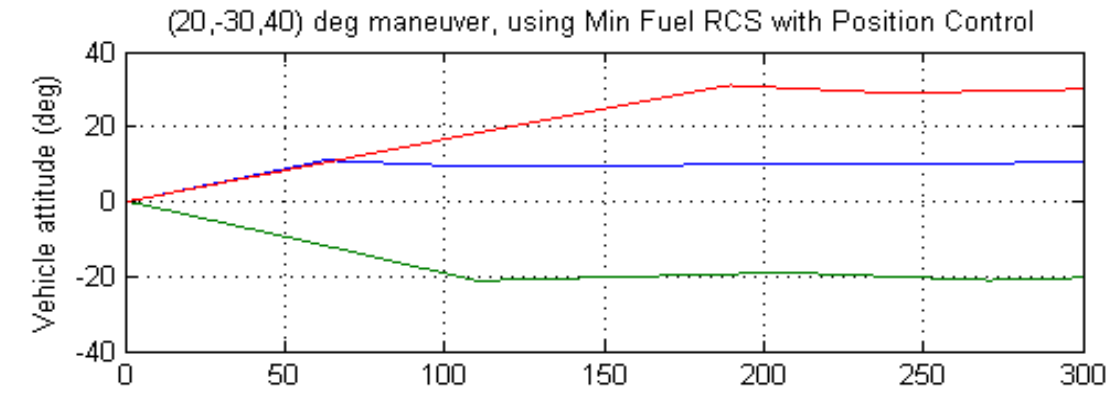
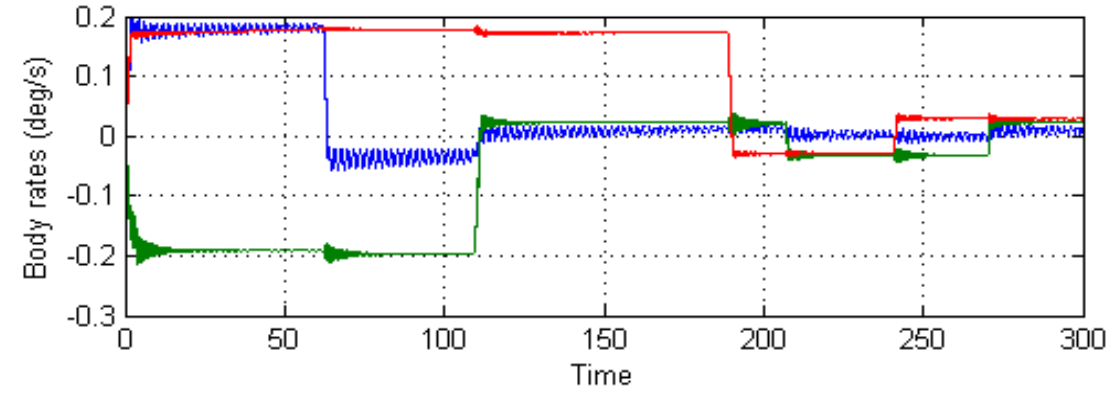


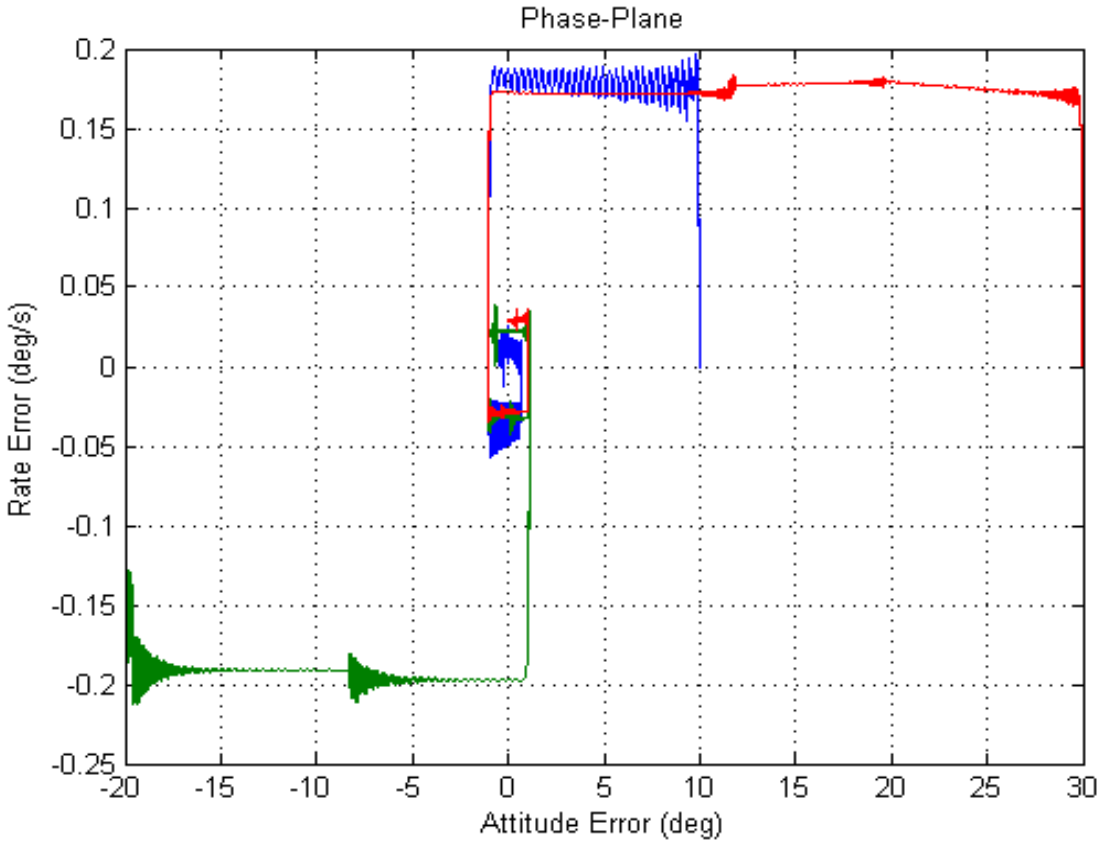
Figure 2.4.7 Flex Spacecraft Discrete Model "*flex_spacecraft_fvp_z.m*" Sampled at 5 msec

The file "start.m" initializes the simulation parameters before running it. The file "pl.m" plots the simulation data, as shown in Figure (2.4.8). The spacecraft attitude responses are similar to the responses obtained from the dot-product ACS logic in Figure (2.2.8). The most interesting result is the fuel usage which is reduced to more than 50% by the fuel minimization logic in comparison to the previous logic, as intended.

Notice, that increasing the sampling period increases also fuel consumption. The minimum fuel logic is constrained to longer control cycles because thrusters have delays and also they cannot be turned on for a very short time. In the comparison shown, the two jet selection algorithms were sampled using the same 0.1 sec period. If the dot-product logic was allowed to be sampled at a rate faster than 0.1 sec, the fuel savings from this fuel optimal method wouldn't be as much. That is, because it is constrained to a slower rate but the other one is not.


(20,-30,40) deg maneuver, using Min Fuel RCS with Position Control
Vehicle attitude (deg)
40
20
0
-20
-40
0
50
100
150
200
250
300


Body rates (deg/s)
0.2
0.1
0
-0.1
-0.2
-0.3
0
50
100
150
200
250
300
Time


Phase-Plane
Rate Error (deg/s)
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-20
-15
-10
-5
0
5
10
15
20
25
30
Attitude Error (deg)

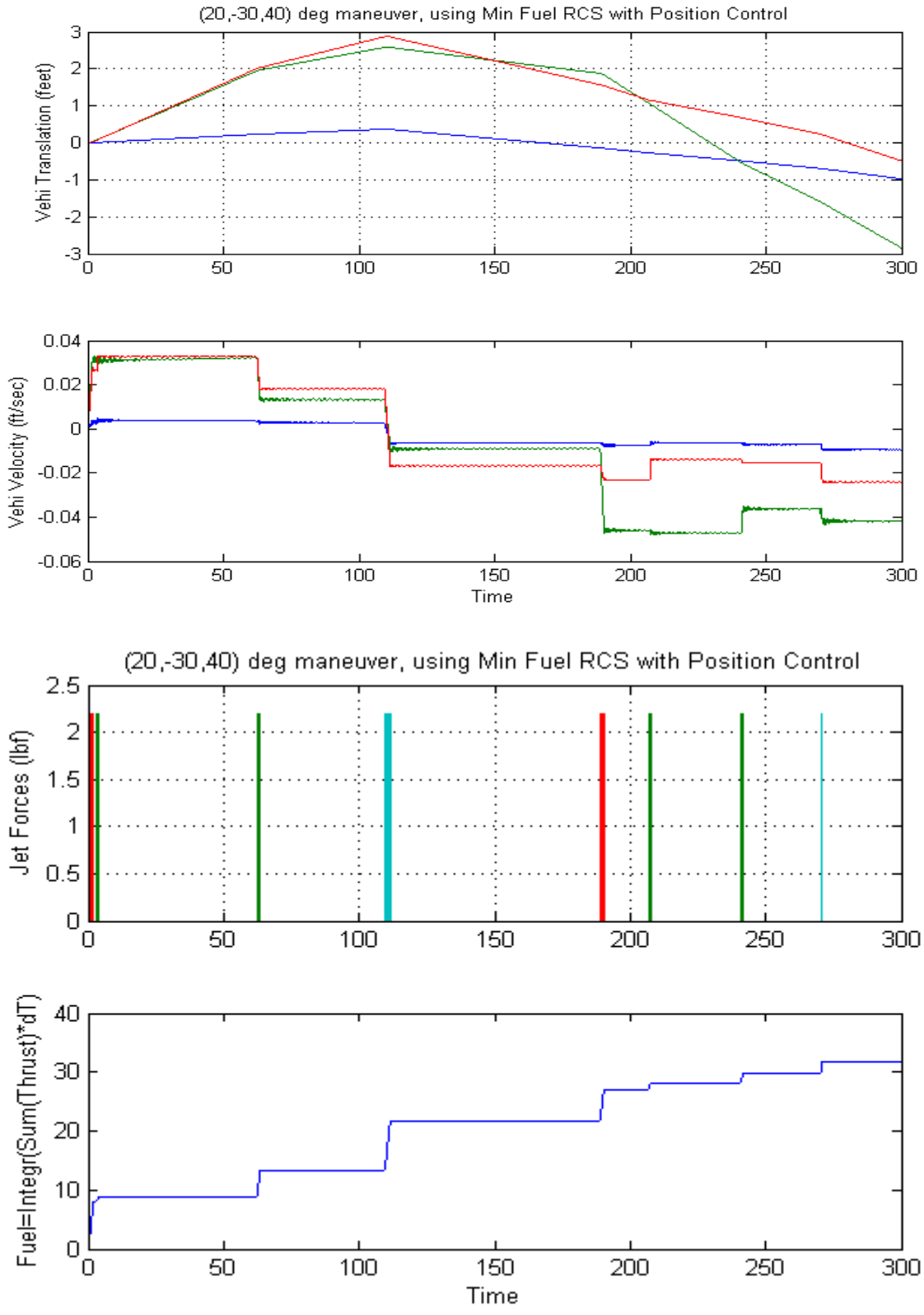


Figure 2.4.8 Spacecraft response to attitude command using the minimum fuel jet-select logic shows significant fuel saving in comparison to the simple dot-product logic

2.5 Extending the Min Fuel Idea to 6-dof Control

The attitude control logic that we have used so far completely ignores the translational position and velocity state of the spacecraft at the end of a maneuver. In some cases we may want to maneuver the spacecraft attitude while maintaining its current position and velocity, without any linear translation, or we may want to simultaneously command attitude and position changes, such as, for example, during docking. In this section we shall extend the min fuel jet selection logic to 6 degrees-of-freedom (6-dof) control for positioning the spacecraft in both rotational and translational directions. The extended 6-dof logic consists of two phase-planes, a rotational phase-plane similar to the one used in section 2.4, and a translational phase-plane that controls spacecraft linear position and operates very similar. The extended 6-dof jet-select logic receives ($\delta\underline{\omega}$) commands from the rotational phase-plane, and ($\delta\underline{v}$) commands from the translational phase-plane. Then it performs a search to identify 6 jets that provide maximum contribution towards the commanded rotational and translational directions combined.

Let us now assume that six jets out of 12 are selected (i, j, k, l, m, and n) and that their corresponding vehicle angular acceleration vectors from each jet are: (a_i , a_j , a_k , a_l , a_m , a_n) respectively. Also the translational acceleration vectors from each jet are: (b_i , b_j , b_k , b_l , b_m , b_n) respectively. The rotational and translational accelerations for a jet (i) are calculated by the following equations

$$\underline{a}_i = J_v^{-1}\left(\underline{d}_i \times \underline{u}_i\right) f_i \qquad \underline{b}_i = f_i \underline{u}_i \Big/ M_v$$

Where:

$\underline{d}_i$ is the moment arm vector of the thruster from the CG
$\underline{u}_i$ is the thruster direction unit vector
f_i is the jet thrust
M_v J_v is the vehicle mass and moment of inertia matrix

If we turn on the 6 selected jets together in the beginning of the control cycle period and leave them on for periods of (t_i , t_j , t_k , t_l , t_m , t_n) respectively, then when the longest firing jet is turned off the vehicle angular and translational rate ($\delta\underline{\omega}$, $\delta\underline{v}$)' is:

$$\begin{bmatrix} \delta\underline{\omega} \\ \delta\underline{v} \end{bmatrix} = \begin{pmatrix} \underline{a}_i & \underline{a}_j & \underline{a}_k & \underline{a}_l & \underline{a}_m & \underline{a}_n \\ \underline{b}_i & \underline{b}_j & \underline{b}_k & \underline{b}_l & \underline{b}_m & \underline{b}_n \end{pmatrix} \begin{bmatrix} t_i \\ t_j \\ t_k \\ t_l \\ t_m \\ t_n \end{bmatrix} = \left[A_{(6x6)}\right]\underline{t} \qquad (2.5.1)$$

If ($\delta\underline{\omega}$, and $\delta\underline{v}$)$_c$ represent the commanded changes in vehicle angular and translational rate at the end of the 0.1 sec cycle, then we can solve for the on-times of the 6 jets by inverting the A matrix.

We assume of course that the jets are properly selected for the commanded directions in order to provide positive on-times, otherwise, it will result into negative on-times.

$$\underline{t} = \left[A^{-1}_{(6x6)}\right]\begin{bmatrix}\delta\underline{\omega}\\ \delta\underline{v}\end{bmatrix}_c \qquad (2.5.2)$$

Assuming that (f_i , f_j , f_k , f_l , f_m , f_n) are the corresponding amounts of fuel flow rates for the six thrusters, the total amount of propellant used by the 6 jets to achieve the commanded ($\delta\underline{\omega}$, and $\delta\underline{v}$)$_c$ is

$$p_{i,j,k,l,m,n} = \begin{pmatrix}\underline{f}_i & \underline{f}_j & \underline{f}_k & \underline{f}_l & \underline{f}_m & \underline{f}_n\end{pmatrix}\left[A^{-1}_{(6x6)}\right]\begin{bmatrix}\delta\underline{\omega}\\ \delta\underline{v}\end{bmatrix} \qquad (2.5.3)$$

The propellant usage factor $p_{i,j,k,l,m,n}$ is the criterion for selecting 6 jets from a total of 12 jets. We also assume that the thrust directions of the jets are properly selected so that there are at least 6 jets which provide a positive torque or force in any commanded direction. For a given commanded ($\delta\underline{\omega}$, and $\delta\underline{v}$)$_c$, the jet-select logic first chooses 6 jets that minimize equation (2.5.3) using function **Jet_Select_6dof**, and it calculates also the "on-times" from equation (2.5.2).

2.5.1 Rotational plus Translational 6-dof Simulation Model, using the Minimum Fuel Jet Selection Logic

The files for this simulation model are in folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis* ***(f) Min Fuel RCS Control 6-dof Flex***". The Matlab file "start.m" initializes the simulation parameters. The Simulink model that implements the fuel optimal 6-dof control logic is "*Sim_Flex-6dof.Mdl*" and it is shown in Figure (2.5.4). It consists of the rotational and translational phase-planes (implemented in functions "*Rotat_PP3.m*" and "*Translat_PP.m*") which operate at 0.1 sec and generate the commands ($\delta\underline{\omega}$, and $\delta\underline{v}$)$_c$ driving the 6-dof jet selection logic. The jet selection logic calls functions "*Jet_main_6dof.m*", "*Jet_Select_6dof.m*", "*Jet_Select_rotat*" and "*Jet_Select-transl*" which select 6 jets and their corresponding "on-times", for each 0.1 sec cycle. The jet forces drive the flex spacecraft dynamics (green block) which is the system "*flex_spacecraft_fvp_z.m*" used earlier and sampled at 5 msec. The inputs to the phase-planes are attitude and translation delta commands.

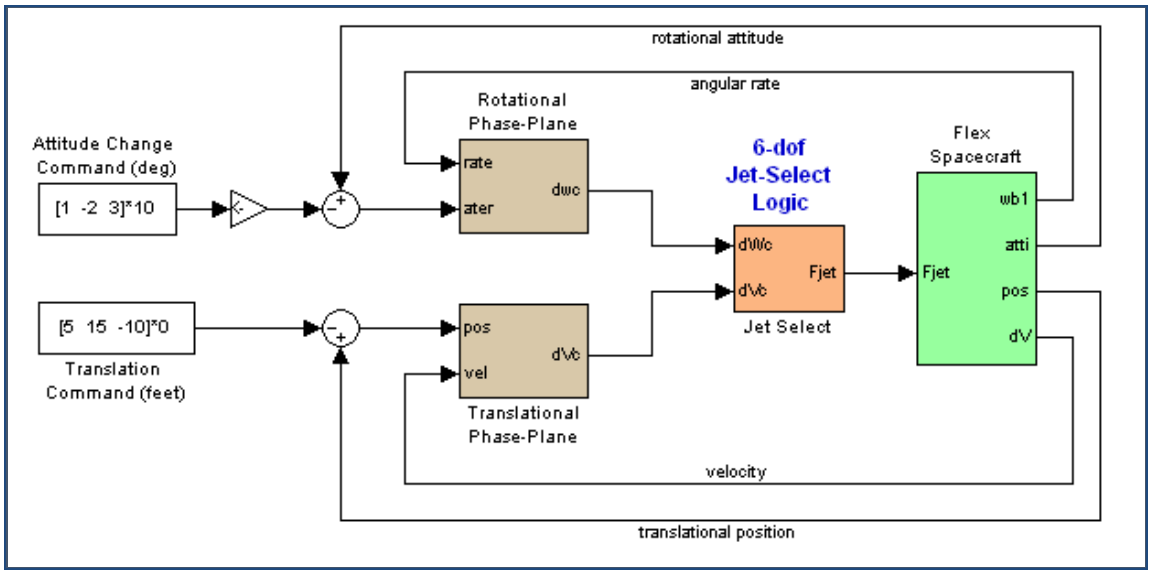


Figure 3.5.4 Minimum Fuel 6-dof Simulation Model "Sim_Flex_6dof.mdl"

In this configuration the RCS controls the vehicle in all 3-translational and 3-rotational directions. The rotational phase-plane issues a change in body rate command and it is implemented in Matlab function "*Rotat_PP3.m*" which. The translational phase-plane issues a change in velocity command and it is implemented in Matlab function "*Translat_PP.m*". Figure (2.5.5) shows the Simulink implementation of the jet selection logic.

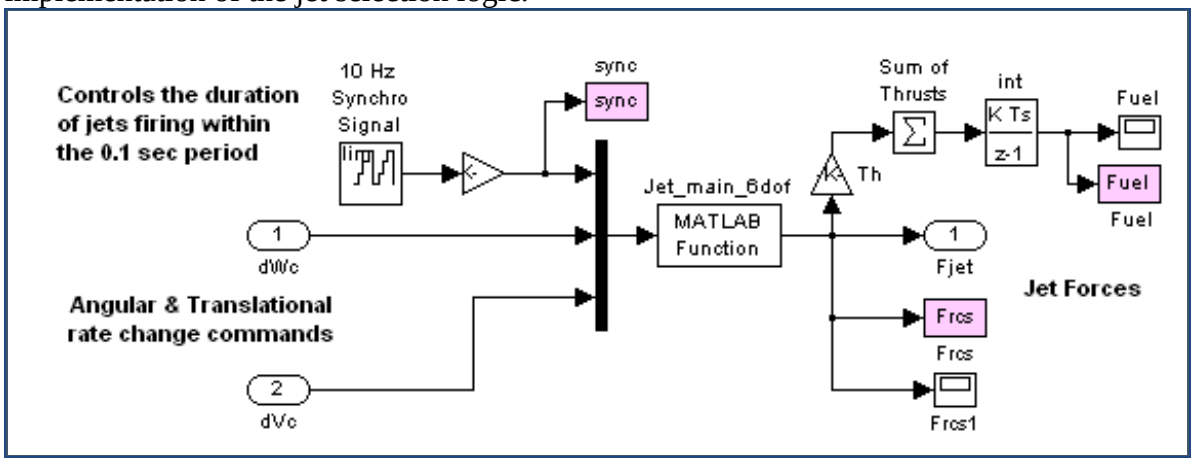
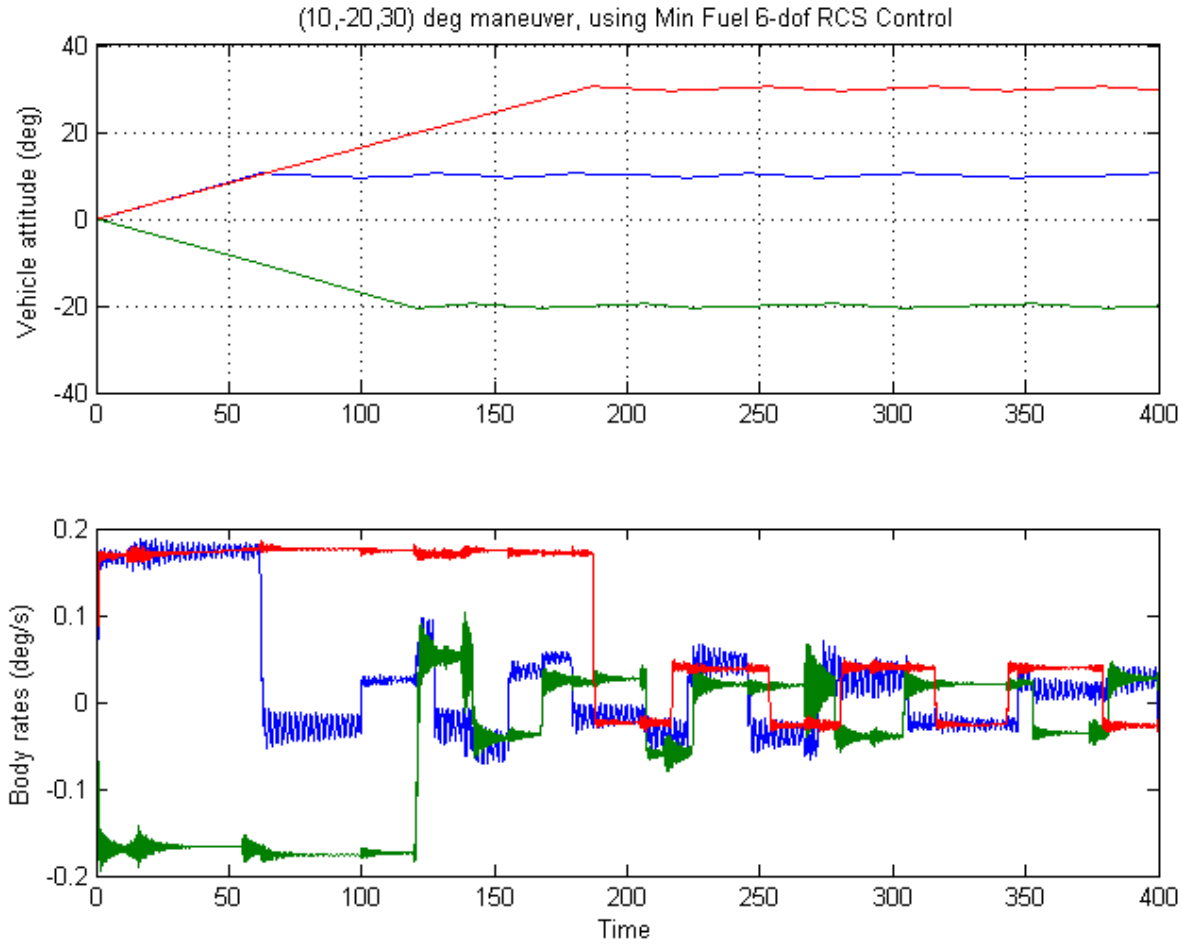


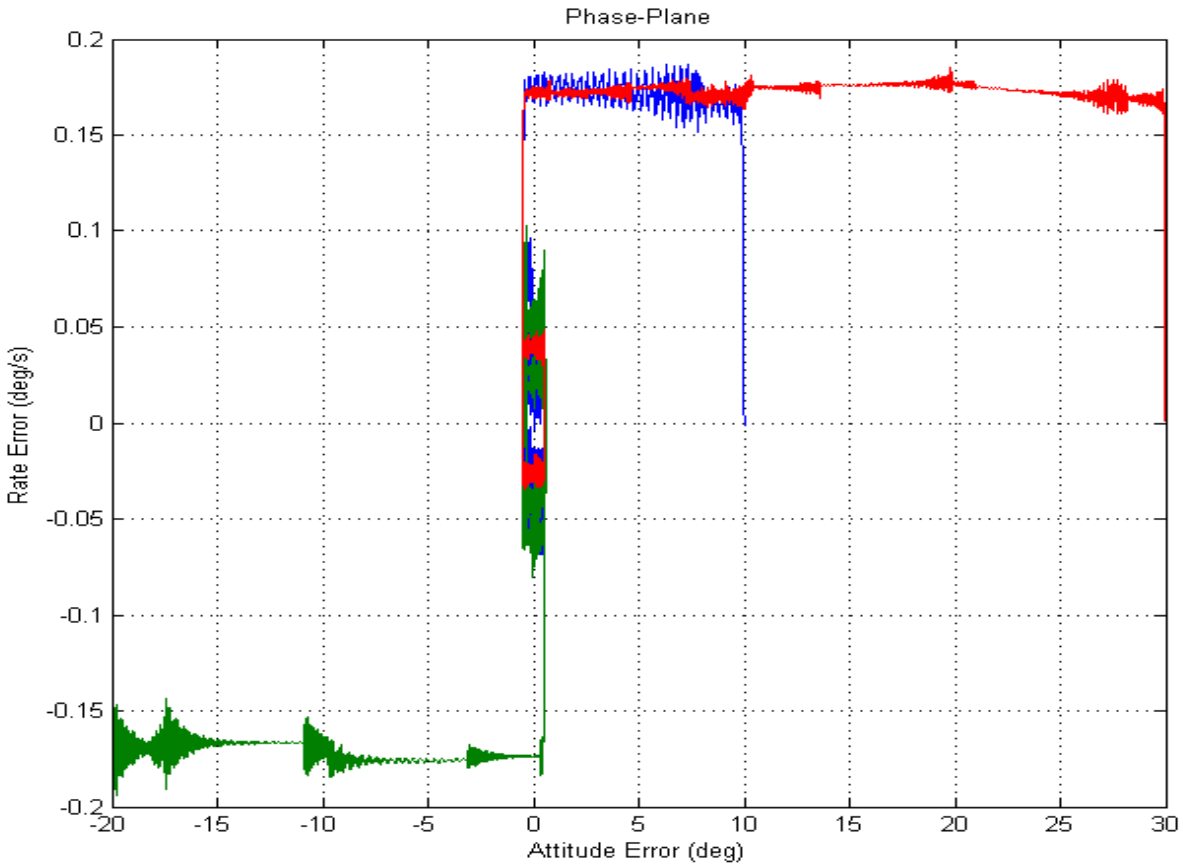
Figure 2.5.5 Minimum Fuel 6-dof Jet Selection Logic

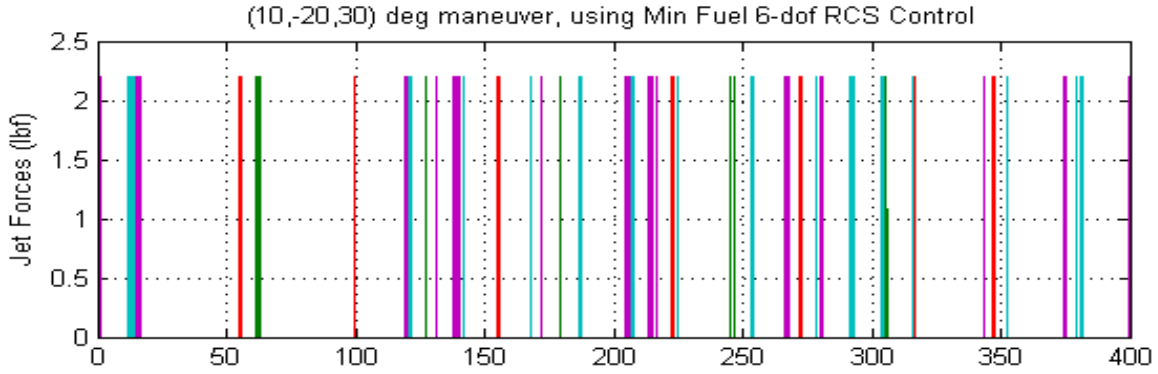
Inside the jet select logic, the change in rate and velocity commands are inputs to Matlab function f= **Jet_main_6dof**(sync, δω, δv), which calculates the jet forces f(i). This function calls the function [js, t]= **Jet_Select_6dof**(δω, δv) which performs a jet search, and selects 6 among the 12 jets that minimize the fuel equation (2.5.3), where (δω, δv) are the change in vehicle rate and velocity commanded by the phase planes. The logic selects 6 jets that best satisfy the combined translational and rotational phase-plane demands. The selected jet numbers are given in the array js(.), and the corresponding on-times in array t(.). The input "sync" is a saw-tooth signal that synchronizes the jet turning-off times in the simulation, not in the actual hardware. In the beginning of the control cycle all 6 selected jets are turned "on". The logic turns them "off" sequentially,

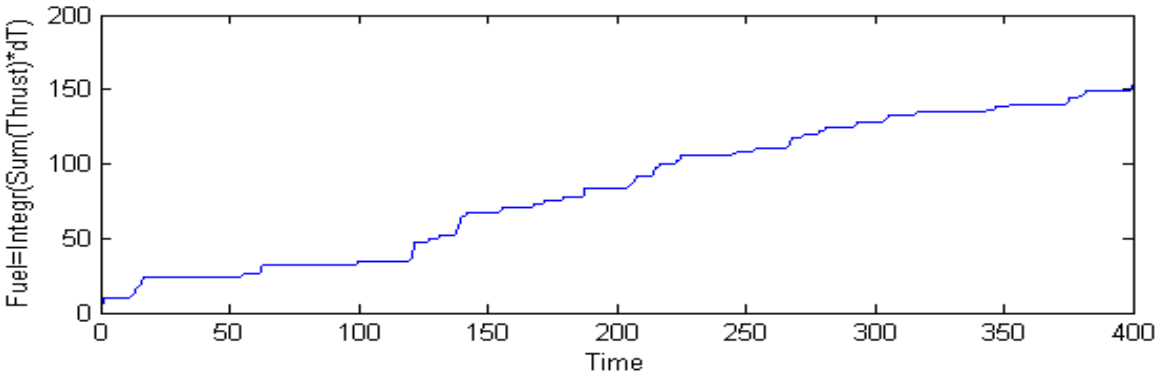
within the 0.1 sec control cycle, according to their on-time values t(i). The cycle repeats every 0.1 second with different jets and on-times. The turning-off times during the control cycle intervals occur in multiples of 5 msec. In the actual hardware implementation each jet receives its "on-time" in the beginning of the 0.1 sec cycle when they get turned-on and it has the logic to turn itself "off" in multiples of 5 msec intervals. The file "start.m" initializes the simulation parameters, and the file "pl.m" plots the simulation data.

Figure (2.5.6) shows attitude and translation simulation results obtained from the 6-dof simulation model "*Sim_Flex_6dof.mdl*". The spacecraft attitude responses are similar to the responses obtained from the 3-dof simulations, but in this case the vehicle does not translate as it did before, but it is holding position while maneuvering in attitude. Of course, this is achieved at the expense of more jet firing and fuel usage.




Phase-Plane
Rate Error (deg/s)
Attitude Error (deg)
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-20
-15
-10
-5
0
5
10
15
20
25
30


(10,-20,30) deg maneuver, using Min Fuel 6-dof RCS Control
Jet Forces (lbf)
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300
350
400


Fuel=Integr(Sum(Thrust)*dT)
Time
200
150
100
50
0
0
50
100
150
200
250
300
350
400

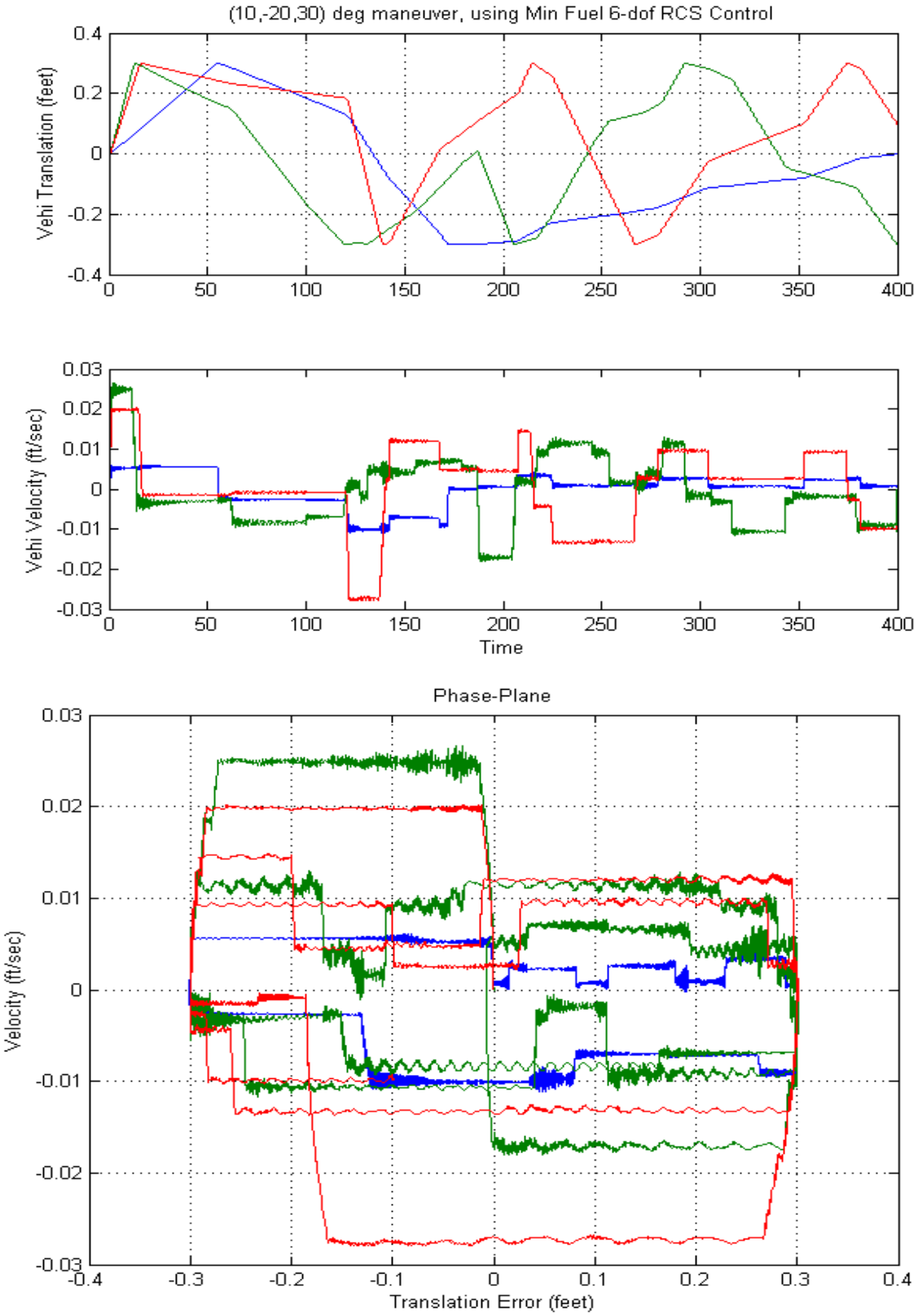


Figure 2.5.6 Attitude maneuver using the 6-dof minimum fuel model, maintains position during maneuvering

2.6 Modeling the Fuel Motion inside the Tank

There is a tank inside the spacecraft that has a spherical shape and contains fuel for the RCS jets. The spacecraft motion causes the fuel to move inside the tank and it is creating disturbance forces on the spacecraft which has the potential to interfere with operations or even cause instability in the control system. There is a need, therefore, to capture the fuel sloshing dynamics in a mathematical model that can be combined with the spacecraft model. Linear spring-mass or pendulum models typically used in launch vehicles are not applicable here because the spacecraft is at zero or very low g and the fuel is not behaving like a linear pendulum. It is reasonable, therefore, to assume that the sloshing fuel will induce a bigger disturbance on the spacecraft when it is lumped together like a soft mass m_s that can slide or bounce against the inner surface of the tank as the vehicle translates and rotates in space, and not when it is spread thin inside the tank.

There are a couple of conceptual models to capture the lumped fuel motion and its reaction forces on the tank, both leading to the same equations. One model assumes that the slosh mass m_s is soft and it behaves like a 3-dimensional elastic pendulum. The mass is attached to a tether, and the other end of the tether is attached to the center of the tank, as if as there is a hook at the center of the tank. To capture the elasticity of the mass we insert a spring between the mass and the tether end. When the string is stretched the reaction forces on the vehicle are applied at the tank center through the tether. The soft mass either slides around the inside surface of the tank with the string stretched or it floats inside the tank when the string is slack, in which case it does not apply any force on the tank. When the mass distance from the tank center exceeds the length of the tether (r) the spring stretches and applies a force at the center through the tether. In another visualization the mass softly splashes against the inner surface of the tank without disintegrating.

There is one additional feature needed to prevent excessive deflections of the slosh mass and to contain it within the walls of the tank. We use a non-linear spring that has a stiffness coefficient $k_s(\delta)$ increasing parabolically with extension (δ), that is $k_s = c_1\delta^2 + c_2$. As the mass approaches the tank surface the radial string force becomes sufficiently high to constrain the mass within the tank walls. So the soft mass can either slide parallel to the surface, or float inside the tank, or bounce against the surface. It can be pictured as shown in Figure (2.6.1), where the fixed length of the string is r, where r is about half the size of the tank radius r_0, and there is a non-linear spring between the end of the string and the center of slosh mass m_s.

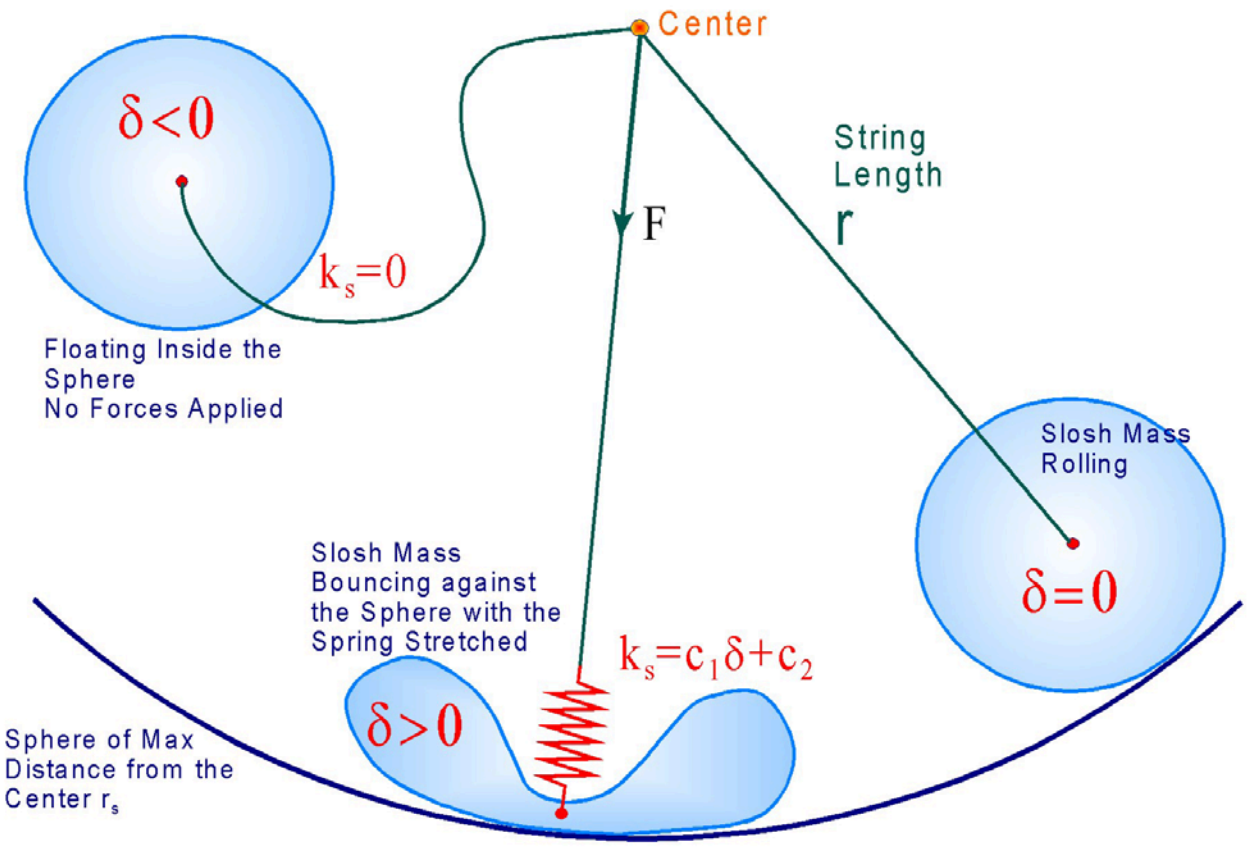
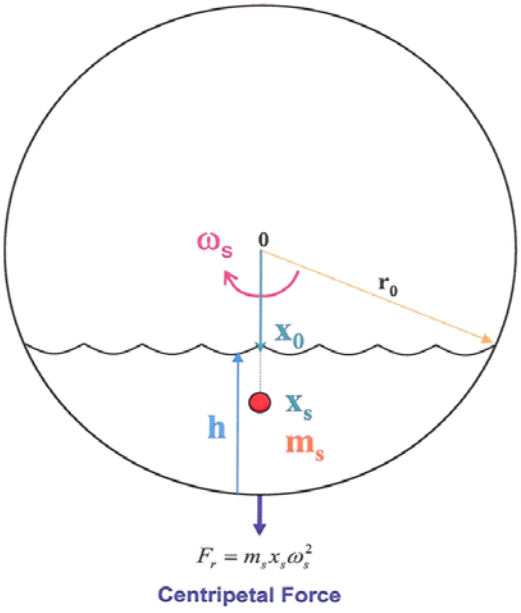


Figure 2.6.1 Conceptual Illustration of the Non-Linear Slosh Pendulum Model

Note, the model parameters, such as the axial and tangential damping coefficients, the length of the tether (r), and the non-linear spring constants are derived from data derived from computational fluid dynamics models. The parameters are adjusted to match the oscillations frequency and rate of decay of the CFD responses. The non-linear spring parameters c_1 and c_2 are adjusted to capture a realistic behavior of the liquid mass as it hits a surface of the tank due to vehicle accelerations.

Another conceptual model that captures the same dynamic effect is to assume that the slosh mass behaves like a soft ball, rolling and bouncing inside a sphere which is approximately half the tank radius, applying forces on the tank perpendicular to the tank surface. Since the tank is spherical the reaction forces always pass through the tank center, same as the soft pendulum model.

2.6.1 What is the worst fuel level?



Before we start analyzing the sloshing problem we must determine: what is the fuel level where the slosh disturbance on the vehicle is maximized. When the tank is almost empty by common sense we know that the sloshing forces are negligible. Also, when the tank is full there is no fuel motion and therefore there is no disturbance. The answer is obviously somewhere in between and we must, therefore, determine what fuel level maximizes the slosh disturbance and use that fuel level in the analysis in order to be on the conservative side.

Let us assume a spherical tank of radius r_0, volume V_0, holding a total fuel mass M_0. The fuel is rotating around the inside surface at an angular rate ω_s held together under the influence of the centripetal force. Its density is (ρ), where:

$$\rho = \frac{M_0}{V_0} = \frac{3M_0}{4\pi r_0^3} \qquad (2.6.1)$$

Assuming that the fuel surface is almost flat, let us calculate the fuel volume as a function of fuel level height (h)

$$V(h) = \pi \int_0^h \left(r_0^2 - x_0^2\right) dh = \pi \int_0^h \left(2r_0 h - h^2\right) dh$$

$$V(h) = \pi h^2 \left(r_0 - \frac{h}{3}\right) \qquad (2.6.2)$$

The fuel mass can be calculated as a function of the fuel surface height or as a function of the surface distance from the tank center x_0.

$$m_s = \pi\rho h^2 \left(r_0 - \frac{h}{3}\right) = \pi\rho \left(\frac{2}{3} r_0^3 + \frac{1}{3} x_0^3 - r_0^2 x_0\right) \qquad (2.6.3)$$

We can also derive equations for the tank fill ratio f_r as a function of fuel height (between zero and one) and the pendulum length l_p between the center of the tank and the fuel center of mass.

$$f_r = \frac{h^2(3r_0 - h)}{4r_0^3} \qquad l_p = \frac{(2r_0 - h)^2}{4\left(r_0 - \frac{h}{3}\right)} \qquad (2.6.4)$$

It seems reasonable to assume that the disturbance on the vehicle will be maxed when the slosh moment of the liquid from the tank center is maximized. The slosh moment is equal to the liquid mass times the distance of its center of mass from the tank center.

$$m_s x_s = \int_{x_0}^{r_0} x\,dm = \pi\rho \int_{x_0}^{r_0} x\left(r_0^2 - x^2\right)dx$$
$$m_s x_s = \frac{\pi\rho}{4}\left(r_0^2 - x_0^2\right)^2 \qquad (2.6.5)$$

The disturbance on the vehicle is maximized when

$$\frac{d}{dx}moment = x_0\left(x_0^2 - r_0^2\right) = 0 \qquad (2.6.6)$$

This happens when x_0=0, that is, when the tank is half full. By combining equations (2.6.3) and (2.6.5) we can solve for the slosh mass distance from the tank center x_s

$$x_s = \frac{\left(r_0^2 - x_0^2\right)^2}{4h^2\left(r_0 - \frac{h}{3}\right)} \qquad (2.6.7)$$

As an alternative approach to calculating max disturbance conditions, let us assume that the liquid mass is spinning inside the tank. It is obvious to assume that the disturbance on the vehicle is maximized when the moment of inertia (I_{slosh}) of the liquid mass about the center of the tank is maximized, where:

$$I_{slosh} = \pi\rho \int_{x_0}^{r_0} x^2\left(r_0^2 - x^2\right)dx = \pi\rho\left[\frac{r_0^2 x^3}{3} - \frac{x^5}{5}\right]_{x_0}^{r_0}$$
$$I_{slosh} = \pi\rho\left[\frac{2r_0^5}{15} + \frac{x_0^5}{5} - \frac{r_0^2 x_0^3}{3}\right] \qquad (2.6.8)$$

The slosh moment of inertia is maximized when

$$\frac{d}{dx}I_{slosh} = x_0^2\left(x_0^2 - r_0^2\right) = 0 \qquad (2.6.9)$$

This happens again when x_0=0, and the fuel height $h = r_0$, that is, the worst disturbance condition is when the tank is half full. In this case the pendulum length, or the slosh mass distance from the center from equation (2.6.7) is

$$r = \frac{3}{8}r_0 \qquad (2.6.10)$$

In our subsequent slosh analysis, therefore, we shall assume that the fuel tank is half full, and the sloshing pendulum mass is equal to half of the full tank mass, and the soft pendulum length (r) is only 3/8 of the actual tank radius (r_0).

2.6.2 Zero-g Non-Linear Slosh Model

The acceleration ($\underline{a}_t$) of the spacecraft at the center of the tank is

$$\underline{a}_t = -\underline{d}_s \times \dot{\underline{\omega}}_b + \underline{a}_s \quad where: \ \underline{d}_s = \underline{l}_{tnk} + \underline{x}_s - \underline{l}_{CG}$$

Where:

$\underline{d}_s$ is the distance of the slosh mass from the spacecraft CG,

$\dot{\underline{\omega}}_b$ is the spacecraft angular acceleration,

$\underline{a}_s$ is the spacecraft translational acceleration at the CG.

When the pendulum string is stretched, the tension at the string F_{st} can be expressed by the following equation

$$\begin{aligned} F_{st} &= m_s(r+\delta)\dot{\theta}^2 && centripetal\ force\ due\ to\ angular\ rate\ of\ mass \\ &+ k_s(\delta)\delta && non-linear\ spring\ force\ due\ to\ deflection\ \delta \\ &+ k_{d1}(\dot{\underline{x}}_s \bullet \underline{u}_1) && axial\ viscous\ friction \end{aligned}$$

Where:

$\dot{\theta}$ is the angular rate of the pendulum relative to tank,

$k_s(\delta)$ is the non-linear spring constant of the sluggish mass which is a function of spring displacement (δ), $k_s = c_1\delta^2 + c_2$

k_{d1} is the axial damping coefficient

$\dot{\underline{x}}_s$ is the slosh mass velocity relative to the tank, dotted with

$\underline{u}_1$ which is the unit vector from the tank center to the slosh mass

$$\underline{u}_1 = \frac{\underline{x}_s}{|\underline{x}_s|}$$

The inertial acceleration of the slosh mass consists of two components, the acceleration of the mass relative to the tank $\ddot{\underline{x}}_s$, plus the inertial acceleration of the tank $\underline{a}_t$, and it is the result of axial forces from the string along $\underline{u}_1$, plus viscous forces as it rotates around rubbing against the inside of the tank along $\underline{u}_2$.

$$m_s(\ddot{\underline{x}}_s + \underline{a}_t) = -F_{st}\underline{u}_1 - (k_{d2}\dot{\theta})\underline{u}_2$$

Where:

k_{dv} is the tangential damping coefficient of the mass as it slides along the surface creating a force parallel to the surface resisting the mass motion along $\underline{u}_2$

$\underline{u}_2$ which is the unit vector parallel to the surface in the velocity direction

$$\underline{u}_v = \frac{\dot{\underline{x}}_s}{|\dot{\underline{x}}_s|} \qquad \underline{u}_2 = [(\underline{u}_1 \times \underline{u}_v) \times \underline{u}_1]$$

The disturbance force on the spacecraft $F_{s/c}$ is equal and opposite to the force on the slosh mass and the torque on the spacecraft is obtained by cross-multiplying $F_{s/c}$ with the distance d_s of the slosh mass from the spacecraft CG.

$$\underline{F}_{s/c} = F_{st}\underline{u}_1 + \left(k_{d2}\dot{\theta}\right)\underline{u}_2$$
$$\underline{T}_{s/c} = \left(\underline{d}_s \times \underline{F}_{s/c}\right) \; where: \underline{d}_s = \underline{l}_{tnk} + \underline{x}_s - \underline{l}_{CG}$$

The velocity and position of the slosh mass m_s with respect to the tank are obtained by integrating the slosh mass acceleration starting from velocity and position initial conditions v_0 and p_0

$$\underline{\dot{x}}_s = \underline{v}_0 + \int_0^t \underline{\ddot{x}}_s(t)\,dt \qquad \underline{x}_s = \underline{p}_0 + \int_0^t \underline{\dot{x}}_s(t)\,dt$$

The slosh mass angular rate $\dot{\theta}$ is obtained by resolving the slosh mass velocity $\underline{\dot{x}}_s$ along the $\underline{u}_2$ direction. The pendulum angle θ is a function of the x-direction component of vector u_1.

$$\dot{\theta} = \left(\frac{\dot{x}_s}{r+\delta}\right) \bullet \underline{u}_2 \qquad \theta = \cos^{-1}\left[u_1(1)\right]$$

2.6.3 Implementing and Testing the Zero-g Slosh Model alone

Before coupling the slosh equations with the spacecraft dynamic model we are going to create a separate zero-g pendulum slosh model "*Slosh.mdl*", shown in Figure (2.6.2). This system will be used to test the slosh dynamics alone under the influence of external forces, starting from an initial condition of m_s. The slosh equations are implemented in Matlab function "*Slosh_0g.m*". The inputs to the Simulink model are: spacecraft rotational acceleration vector, and translational acceleration vector at the CG, coming from the vehicle dynamic model. The spacecraft accelerations induce forces on the mass and opposite reaction forces on the vehicle. The slosh mass acceleration relative to tank $\underline{\ddot{x}}_s$ is integrated twice to calculate the mass velocity and position relative to the tank. The model output is the reaction force vector which is applied to the vehicle model at the center of the tank. The forces are applied only when the spring is stretched, that is, $\delta > 0$. The file "*start.m*" loads the tank mass properties and initializes the mass position and velocity relative to the tank. The file "*plsl.m*" plots the simulation parameters.

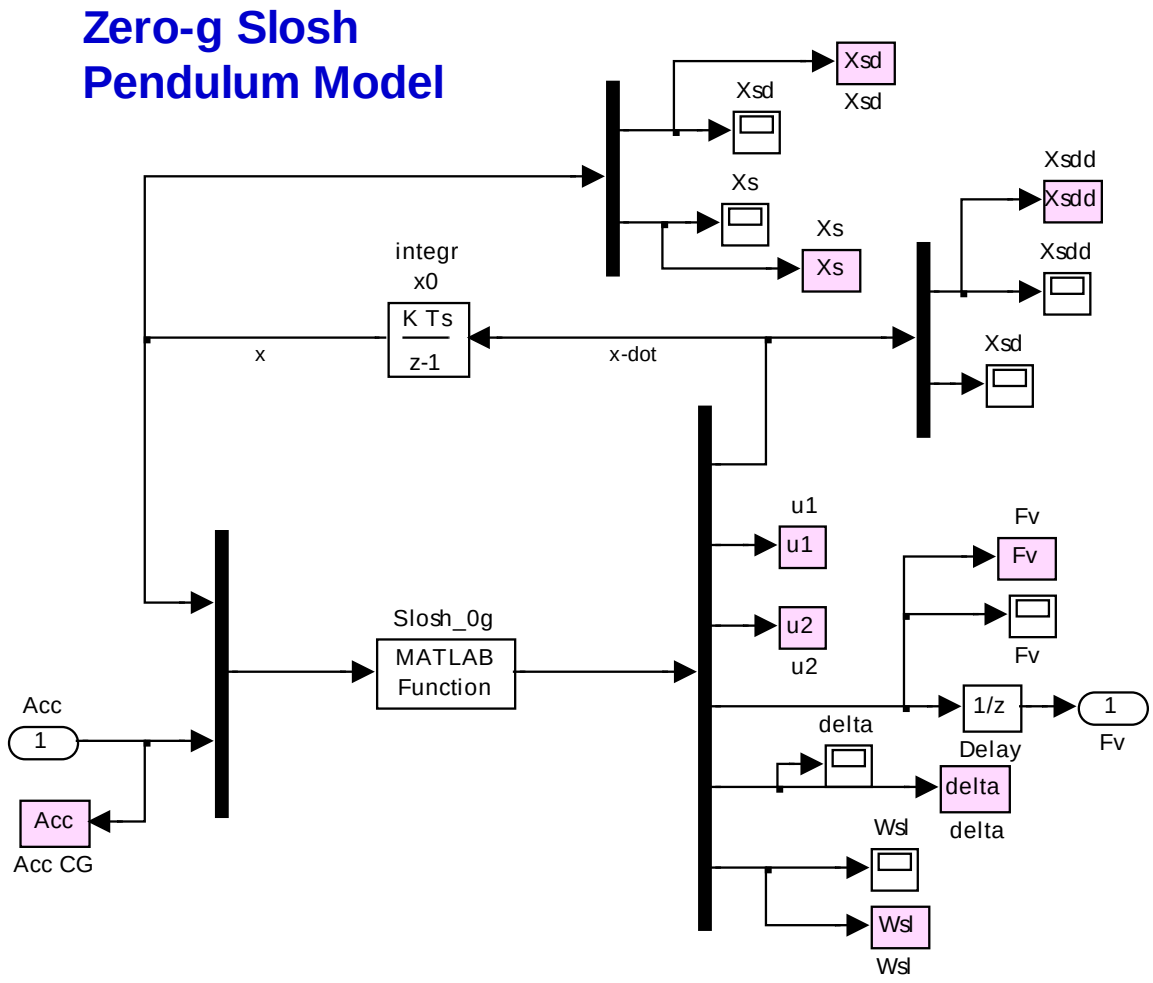


Figure 2.6.2 Zero-g Pendulum Slosh Model "Slosh.Mdl" that uses Matlab function "Slosh_0g.m"

Figure (2.6.3) shows the slosh model response when there are no external accelerations, but the mass is initially placed in the extreme +y position while at the same time it has a velocity of 0.5 (ft/sec) in the –x direction. So it rolls inside the tank surface in the x-y plane while it is slowing down due to surface damping. As it slows down it applies less reaction force on the tank.

Figure (2.6.4) shows another situation where the mass is in the extreme +y position and it has no velocity, but the spacecraft is accelerating in the +x direction. This causes the mass to roll inside the tank and oscillate in ±y direction passing through zero. Eventually, damping causes the slosh mass to slow down and to converge in the –x direction. Also, the force on the vehicle ends up in the –x direction. Notice, that the oscillation in the x direction is twice the frequency of the y oscillation. This model will now be combined with the flex vehicle model.

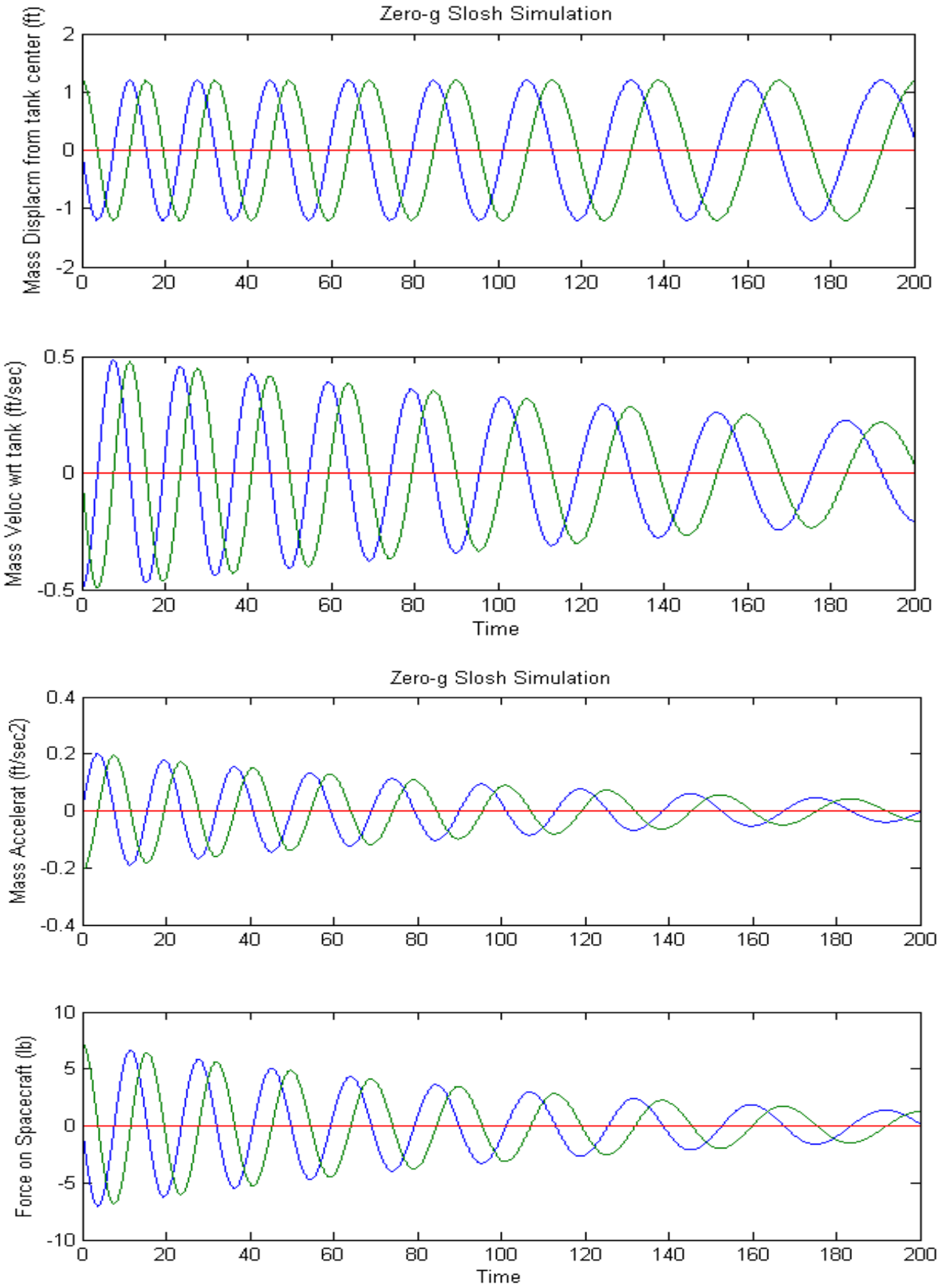


Figure 2.6.3 Slosh mass rolling inside the tank due to initial velocity

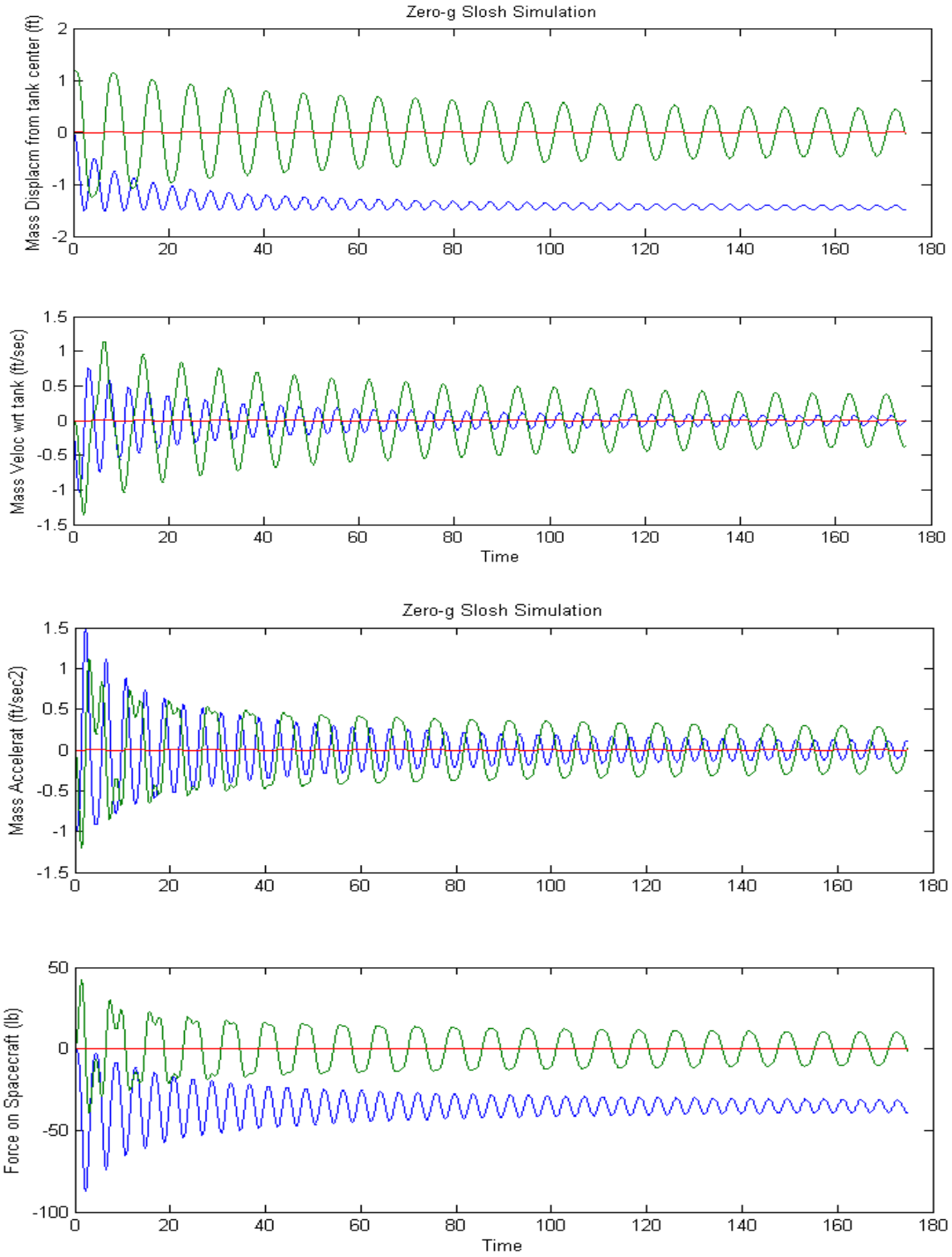


Figure 2.6.4 Slosh mass oscillations under constant +x acceleration

2.7 Minimum Fuel RCS 6-dof Simulation with Fuel Sloshing and Flexibility at Zero-g

Now that we have developed our rigid-body and flex spacecraft models, and we have successfully analyzed the fuel minimization algorithm in both rotational and translational directions, and also tested the zero-g slosh model, the next step is to integrate all these models together in a 6-dof simulation that will be used to analyze the simultaneous, attitude and translation control while sloshing. Actually, we are going to develop two integrated models, a linear, and a non-linear model for comparison. The simulation files in this analysis are located in folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis**__*(h) NonLin RB+Slosh+Flex 6-dof RCS Attitude Control*__". The file "*start.m*" initializes the spacecraft parameters. The slosh mass is initialized with an initial position x_{s0}, and velocity x_{sd0} relative to the center of the tank.

2.7.1 Linear 6-dof Model with Slosh and Flex

The linear Simulink model is in file "*Sim_Lin_Flex_Slo_6dof.mdl*", and shown in Figure (2.7.1). For spacecraft dynamics it uses the discrete state-space model file "*flex-spacecraft-fem_z.m*", title: "*RB+Flex Spacecraft with RCS and CMG (Z-Transf)*", which has 6 rigid-body modes and 40 flex modes, and was created using the flex spacecraft modeling program.

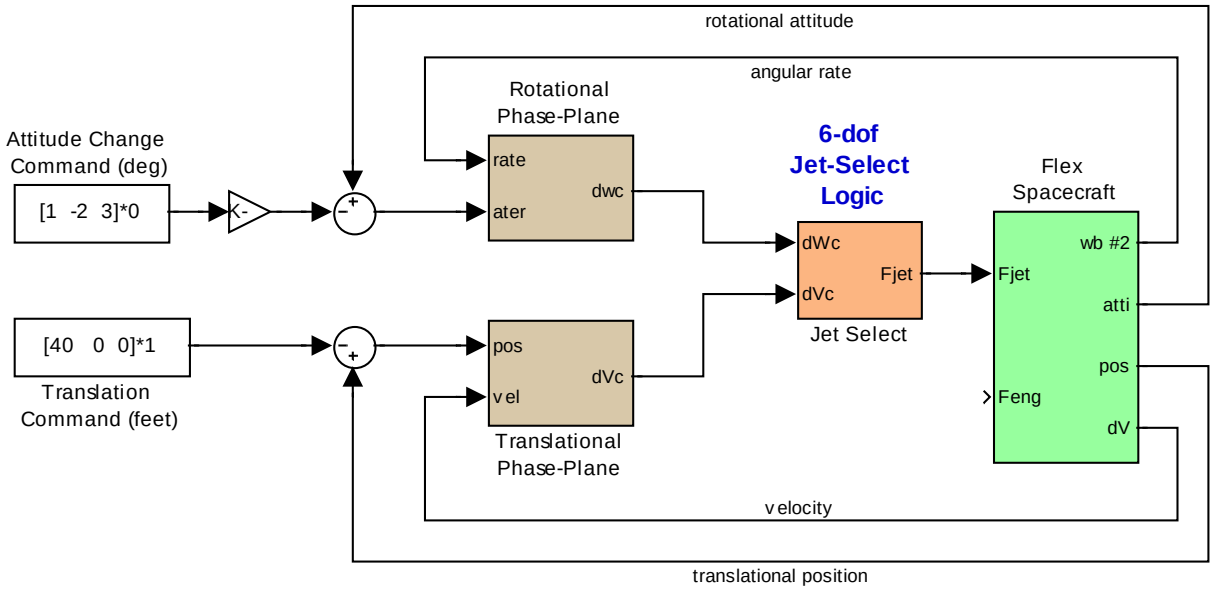
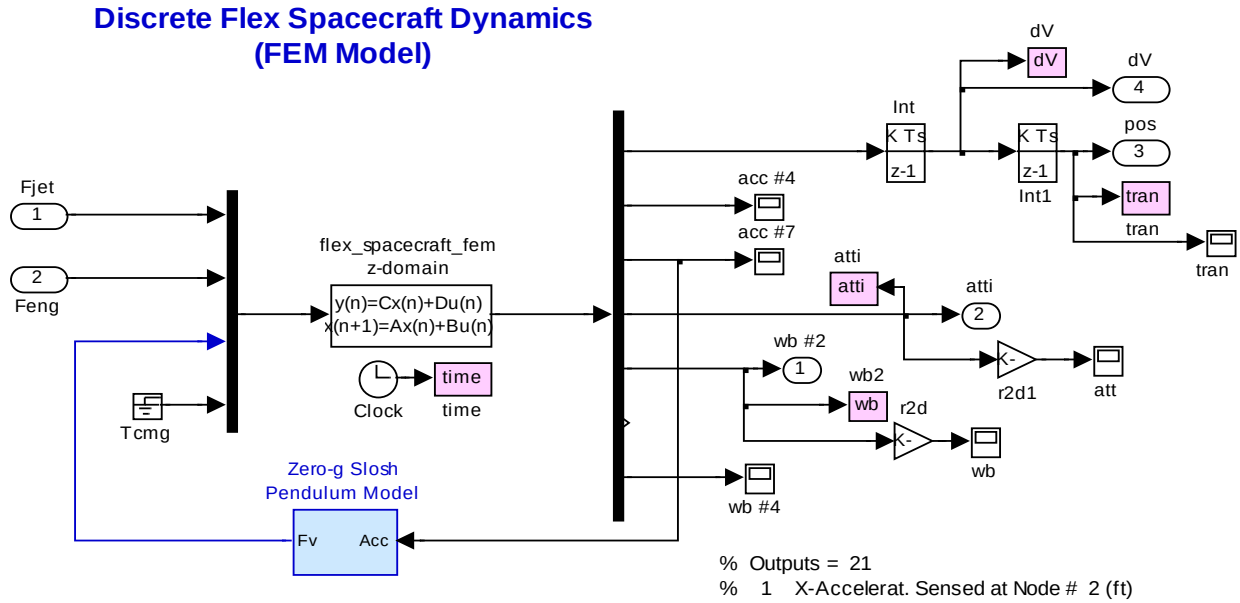


Figure 2.7.1 Min Fuel 6-dof Simulation Model "Sim_Lin_Flex_Slo_6dof.mdl" that includes Slosh and Flexibility

The model uses the two separate rotational and translational phase-planes and the 6-dof fuel minimizing jet selection logic that was described in Section (2.5). The simulation is running at 5 msec, and the phase-planes are running at 100 msec. Figure (2.7.2) shows the spacecraft model in

detail. The zero-g slosh model is shown closing a mechanical feedback loop between the vehicle acceleration at the center of the tank and the force applied at the tank center. The slosh model and other spacecraft parameters are initialized by file "start.m". The fuel slosh block uses the Matlab function "**f=Slosh_0g(Acc)**" which calculates the reaction forces applied at the tank by the slosh mass as a function of the tank accelerations. The simulation is initialized automatically, and when it completes it runs file "*pl2.m*" to plot the results. Figure (2.7.3) shows the results from an attitude maneuver. The position is constrained to within one foot of error while maneuvering.




```
% Inputs  = 19
%   1    Force  No  1 Applied at Node #  8 (lbf)
%   2    Force  No  2 Applied at Node #  9 (lbf)
%   3    Force  No  3 Applied at Node # 10 (lbf)
%   4    Force  No  4 Applied at Node # 11 (lbf)
%   5    Force  No  5 Applied at Node # 12 (lbf)
%   6    Force  No  6 Applied at Node # 13 (lbf)
%   7    Force  No  7 Applied at Node # 14 (lbf)
%   8    Force  No  8 Applied at Node # 15 (lbf)
%   9    Force  No  9 Applied at Node # 16 (lbf)
%  10    Force  No 10 Applied at Node # 17 (lbf)
%  11    Force  No 11 Applied at Node # 18 (lbf)
%  12    Force  No 12 Applied at Node # 19 (lbf)

%  13    Force  No 13 Applied at Node #  5 (lbf)

%  14    Force  No 14 Applied at Node #  7 (lbf)
%  15    Force  No 15 Applied at Node #  7 (lbf)
%  16    Force  No 16 Applied at Node #  7 (lbf)

%  17    Torque No  1 Applied at Node #  6 (ft-lb)
%  18    Torque No  2 Applied at Node #  6 (ft-lb)
%  19    Torque No  3 Applied at Node #  6 (ft-lb)
```

```
% Outputs = 21
%   1    X-Accelerat. Sensed at Node #  2 (ft)
%   2    Y-Accelerat. Sensed at Node #  2 (ft)
%   3    Z-Accelerat. Sensed at Node #  2 (ft)

%   4    X-Accelerat. Sensed at Node #  4 (ft)
%   5    Y-Accelerat. Sensed at Node #  4 (ft)
%   6    Z-Accelerat. Sensed at Node #  4 (ft)

%   7    X-Accelerat. Sensed at Node #  7 (ft)
%   8    Y-Accelerat. Sensed at Node #  7 (ft)
%   9    Z-Accelerat. Sensed at Node #  7 (ft)

%  10    X-Rotation   Sensed at Node #  2 (radian)
%  11    Y-Rotation   Sensed at Node #  2 (radian)
%  12    Z-Rotation   Sensed at Node #  2 (radian)

%  13    X-Rot. Rate  Sensed at Node #  2 (radian)
%  14    Y-Rot. Rate  Sensed at Node #  2 (radian)
%  15    Z-Rot. Rate  Sensed at Node #  2 (radian)

%  16    X-Rot. Rate  Sensed at Node #  1 (radian)
%  17    Y-Rot. Rate  Sensed at Node #  1 (radian)
%  18    Z-Rot. Rate  Sensed at Node #  1 (radian)

%  19    X-Rot. Rate  Sensed at Node #  4 (radian)
%  20    Y-Rot. Rate  Sensed at Node #  4 (radian)
%  21    Z-Rot. Rate  Sensed at Node #  4 (radian)
```

Figure 2.7.2 Spacecraft Dynamics consists of the flex system "flex_spacecraft_fem_z.m" and the Slosh Model Function "Slosh_0g.m" closing a mechanical feedback loop

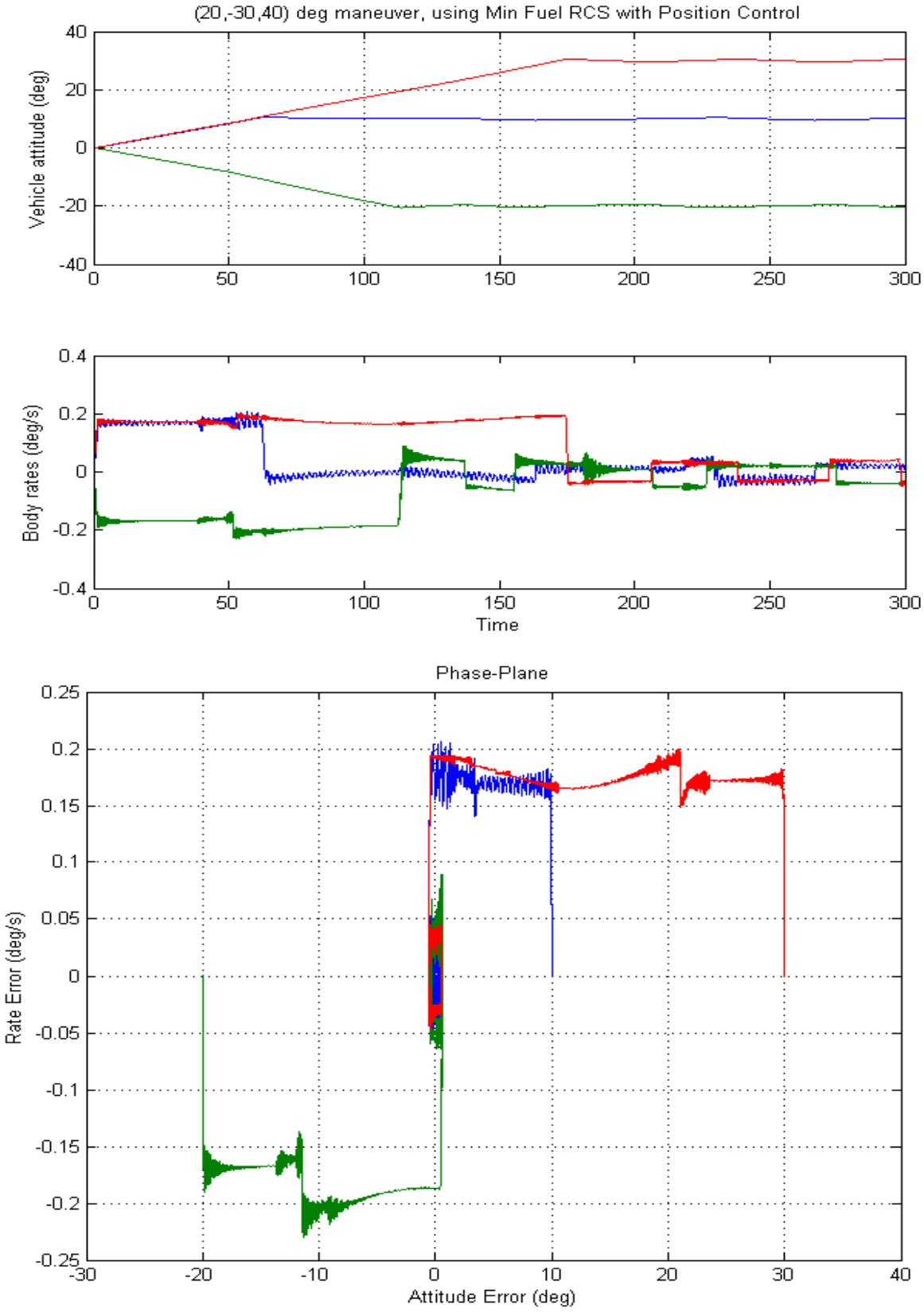


Figure 2.7.2 Performs attitude maneuver as expected

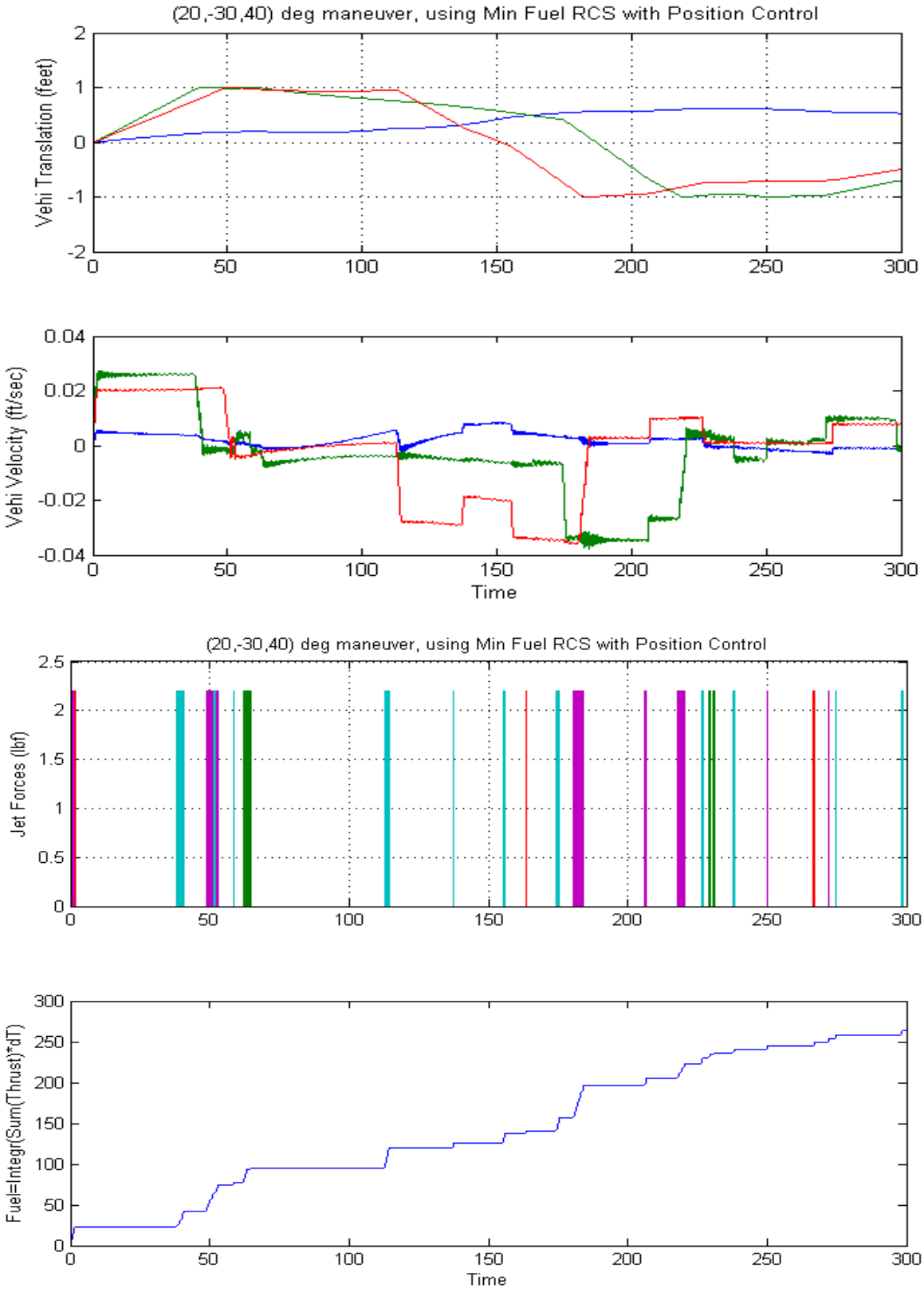


Figure 2.7.2 Position error does not exceed one foot

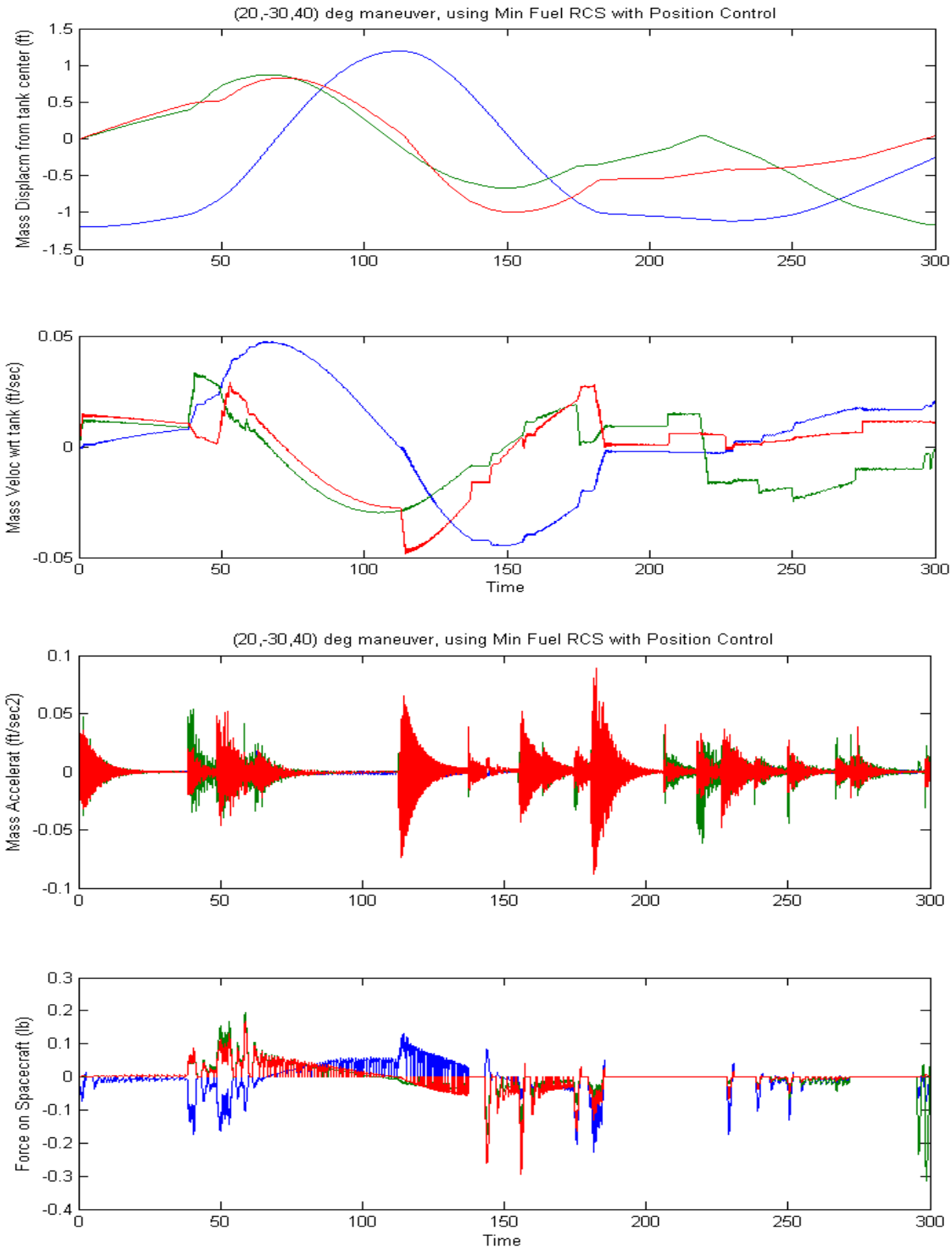


Figure 2.7.2 Slosh mass activity relative to tank and slosh force on the vehicle

2.7.2 Non-Linear 6-dof Model with Slosh and Flex

The non-linear Simulink model is in file "*Sim_NonLin_Flex_Slo_6dof.mdl*", shown in Figure (2.7.3). This model uses quaternion for attitude control and calculates a quaternion error for attitude feedback. The quaternion command (left) is calculated by combining the commanded angle of rotation and the rotation axis, see Figure (2.7.4). The attitude feedback signal (q_e) consists of the 3-dimensional vector part of the quaternion error. For small angles q_e= half the attitude error vector.

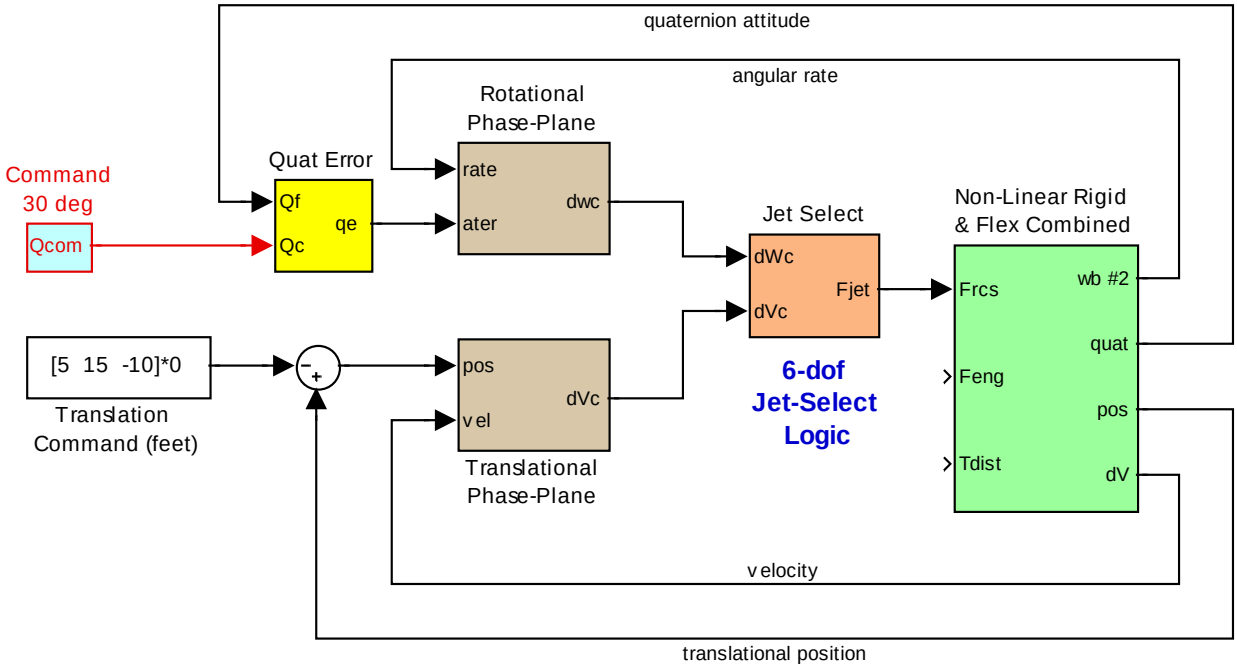


Figure 2.7.3 Min Fuel 6-dof Simulation Model "Sim_NonLin_Flex_Slo_6dof.mdl" that includes Slosh and Flexibility

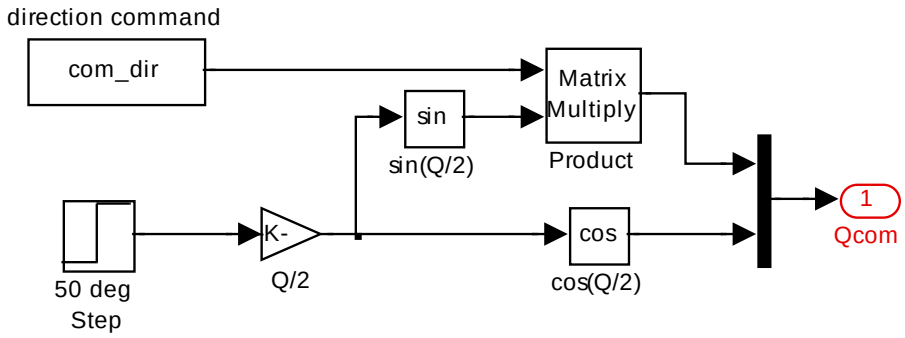
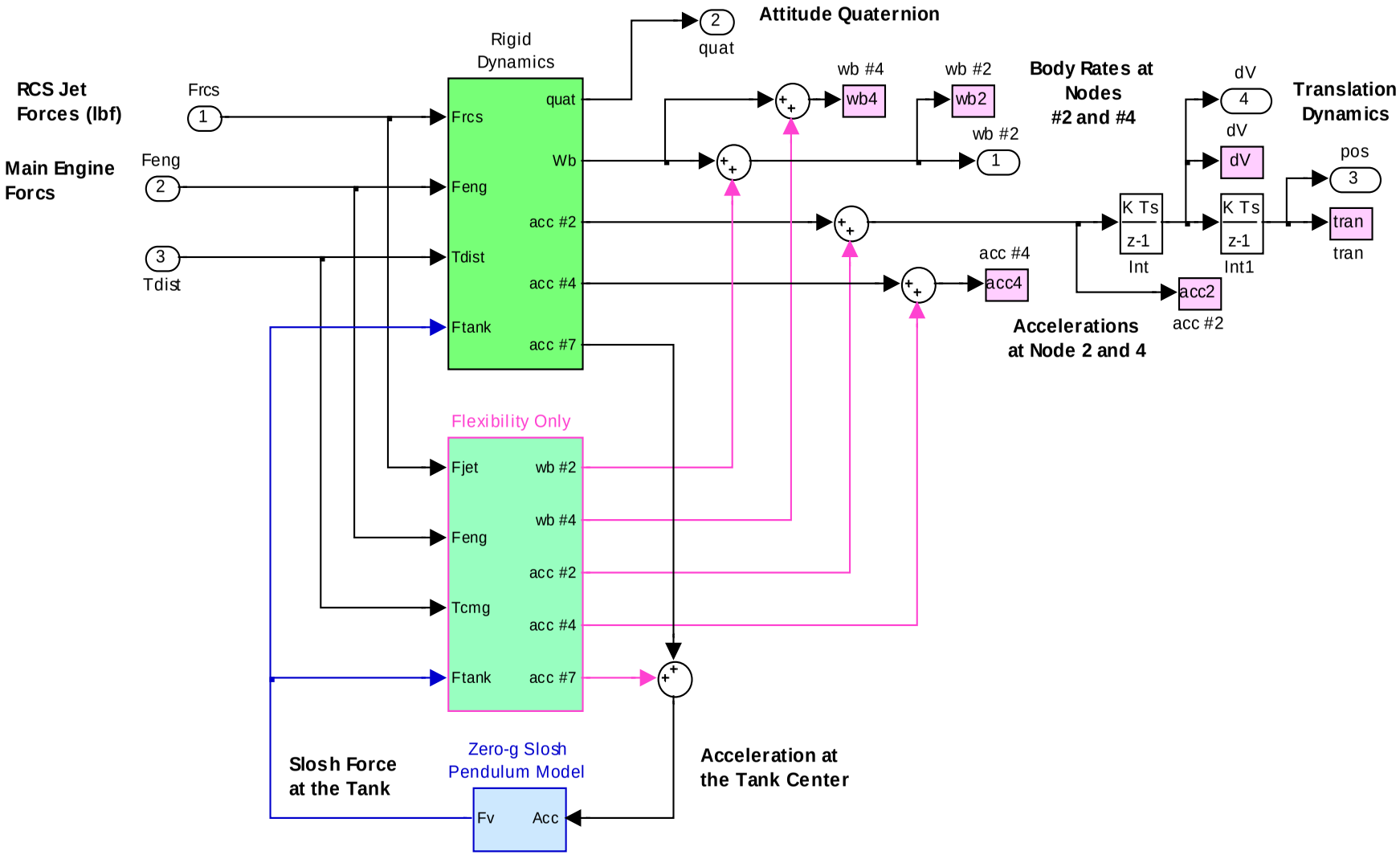


Figure 2.7.4 Quaternion Command Generator

The spacecraft dynamics is shown in Figure (2.7.5). It consists of two dynamic systems in parallel with the slosh model in the feedback path. It uses the 6-dof non-linear rigid-body dynamics which is implemented in Matlab function "*RB_Dynamics.m*". From the forces and moments this function calculates the velocities, angular rates, and integrates the quaternion from the body rates. It calculates also vehicle accelerations at three locations: the navigation base, at one of the solar array joints, and at the center of the fuel tank.
The flexibility model is connected in parallel with the rigid-body model. The flex model uses the state-space system file is: "*flex_only_fem_z.m*". This system is discretized at 5 msec, it has no rigid modes, but consists only of 40 flex modes starting from mode #7. It was created using the flex spacecraft modeling program, and its title is "*Flex Only Spacecraft with RCS and CMG (Z-Transf)*". Both spacecraft models are excited with the same forces and torques. Their outputs are added together and the output signals consist of non-linear spacecraft responses with flexibility superimposed.

The slosh model is also included in the feedback loop, connected between the vehicle acceleration at the center of the tank and the input force at the tank. The slosh model uses the Matlab function "*Slosh_0g.m*" which calculates the reaction forces applied by the slosh mass as a function of the spacecraft accelerations at the tank. The simulation is initialized by file "start.m" and when it completes it plots the results using file "*pl.m*".

Non-Linear Rigid + Flex + Fuel Sloshing Dynamic Model



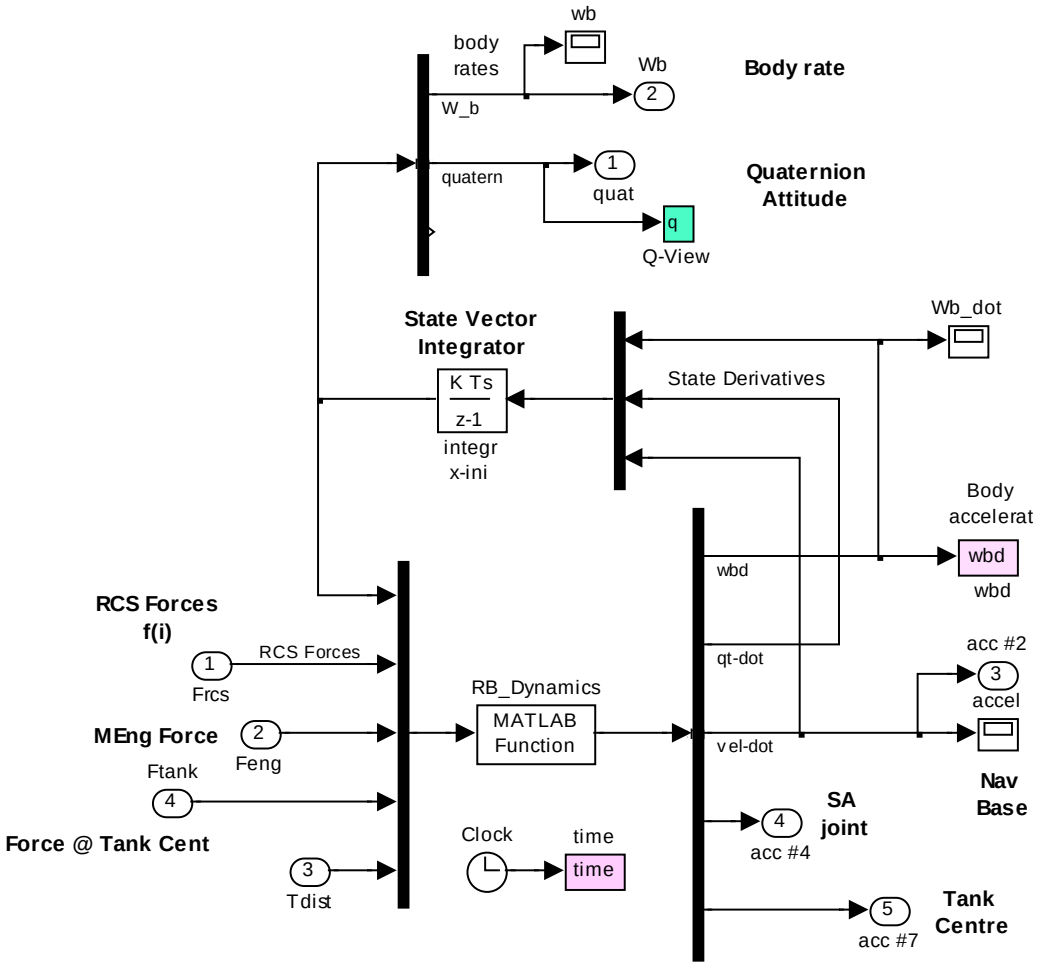
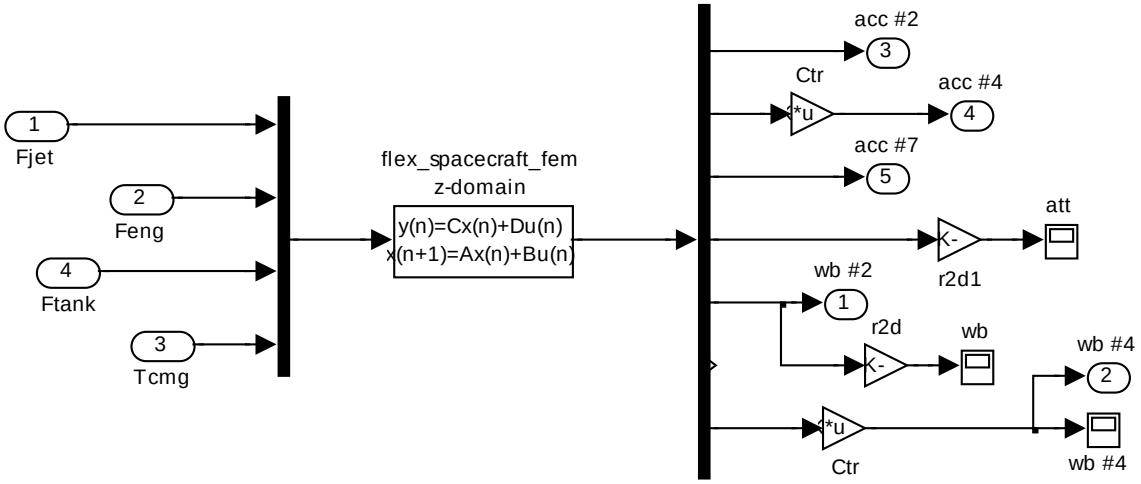


Figure 2.7.6 Rigid-Body Subsystem or the Spacecraft Dynamics Block

Figure (2.7.6) shows the rigid-body subsystem block in detail. It uses the Matlab function "*RB_Dynamics.m*" which contains the non-linear rigid dynamics and integrates the input forces and torques into body rates and accelerations. In addition to the jet and main engine force inputs, the force on the vehicle due to the slosh mass motion, created by the slosh model, is also an input. The outputs are: body rate, attitude quaternion, and spacecraft accelerations at three locations, the navigation base, a sensitive location at the solar array, and at the center of the tank.

The flexibility model is shown in Figure (2.7.7). It uses the discrete (5 msec) state-space system: "*Flex Only Spacecraft with RCS and CMG (Z-Transf)*" loaded from file: "*flex_only_fem_z.m*", which was created in Section 1.1. It contains only the 40 flex modes starting from mode #7. It is excited by the same input forces and torques as the rigid model, but its outputs consist only of flexibility components without rigid-body content. Its outputs consist of accelerations at node numbers 2, 4, and 7, angular rates at nodes 2, 1, and 4, and rotations at node #2. Note that, the acceleration and the rate measurements at node #4 (which is at the left solar array) are in solar array and not in vehicle coordinates, and they are transformed to body coordinates by multiplying them with a transformation matrix "Ctr", so that they can be combined with the rigid-body components from the rigid system.



% Inputs = 19
% 1 Force No 1 Applied at Node # 8 (lbf)
% 2 Force No 2 Applied at Node # 9 (lbf)
% 3 Force No 3 Applied at Node # 10 (lbf)
% 4 Force No 4 Applied at Node # 11 (lbf)
% 5 Force No 5 Applied at Node # 12 (lbf)
% 6 Force No 6 Applied at Node # 13 (lbf)
% 7 Force No 7 Applied at Node # 14 (lbf)
% 8 Force No 8 Applied at Node # 15 (lbf)
% 9 Force No 9 Applied at Node # 16 (lbf)
% 10 Force No 10 Applied at Node # 17 (lbf)
% 11 Force No 11 Applied at Node # 18 (lbf)
% 12 Force No 12 Applied at Node # 19 (lbf)

% 13 Force No 13 Applied at Node # 5 (lbf)

% 14 Force No 14 Applied at Node # 7 (lbf)
% 15 Force No 15 Applied at Node # 7 (lbf)
% 16 Force No 16 Applied at Node # 7 (lbf)

% 17 Torque No 1 Applied at Node # 6 (ft-lb)
% 18 Torque No 2 Applied at Node # 6 (ft-lb)
% 19 Torque No 3 Applied at Node # 6 (ft-lb)

% Outputs = 21
% 1 X-Accelerat. Sensed at Node # 2 (ft)
% 2 Y-Accelerat. Sensed at Node # 2 (ft)
% 3 Z-Accelerat. Sensed at Node # 2 (ft)

% 4 X-Accelerat. Sensed at Node # 4 (ft)
% 5 Y-Accelerat. Sensed at Node # 4 (ft)
% 6 Z-Accelerat. Sensed at Node # 4 (ft)

% 7 X-Accelerat. Sensed at Node # 7 (ft)
% 8 Y-Accelerat. Sensed at Node # 7 (ft)
% 9 Z-Accelerat. Sensed at Node # 7 (ft)

% 10 X-Rotation Sensed at Node # 2 (radian)
% 11 Y-Rotation Sensed at Node # 2 (radian)
% 12 Z-Rotation Sensed at Node # 2 (radian)

% 13 X-Rot. Rate Sensed at Node # 2 (radian)
% 14 Y-Rot. Rate Sensed at Node # 2 (radian)
% 15 Z-Rot. Rate Sensed at Node # 2 (radian)

% 16 X-Rot. Rate Sensed at Node # 1 (radian)
% 17 Y-Rot. Rate Sensed at Node # 1 (radian)
% 18 Z-Rot. Rate Sensed at Node # 1 (radian)

% 19 X-Rot. Rate Sensed at Node # 4 (radian)
% 20 Y-Rot. Rate Sensed at Node # 4 (radian)
% 21 Z-Rot. Rate Sensed at Node # 4 (radian)

Figure 2.7.7 Structural Flexibility Subsystem

The zero-g non-linear pendulum model block is shown in Figure (2.7.8). The slosh mass is initialized at some initial location relative to the tank and it is excited into motion by the vehicle acceleration due to engine firing or the reaction jet thrusts. When its distance from the center exceeds the pendulum length the mass applies forces (F_{tank}) at the tank center, closing the mechanical feedback loop between vehicle acceleration and the input force at the tank in Figure (2.7.5).

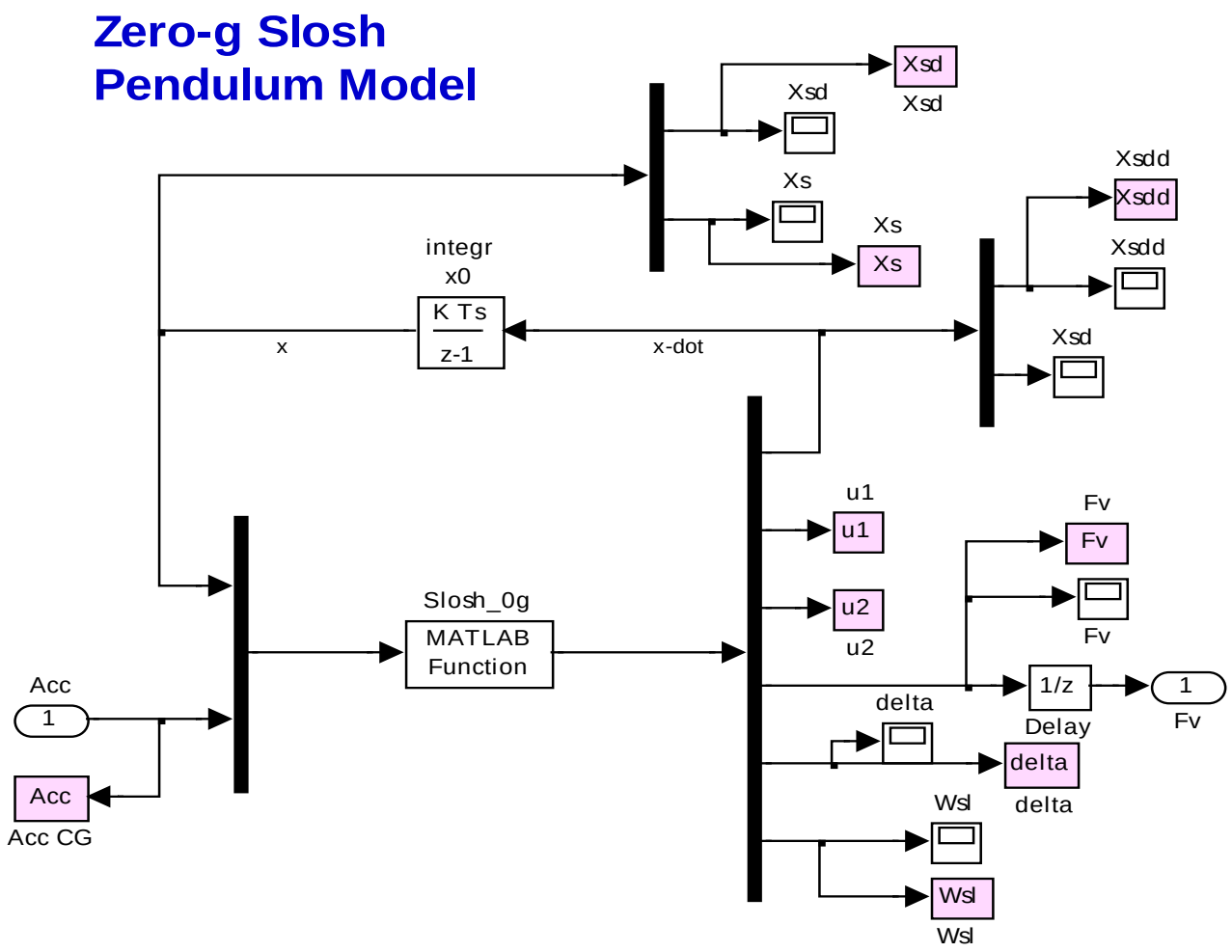


Figure 2.7.8 Zero-gravity Non-Linear Slosh Dynamics Subsystem

The simulation results in Figure (2.7.9) show the spacecraft response to a 50 (deg) attitude command in an arbitrary direction.

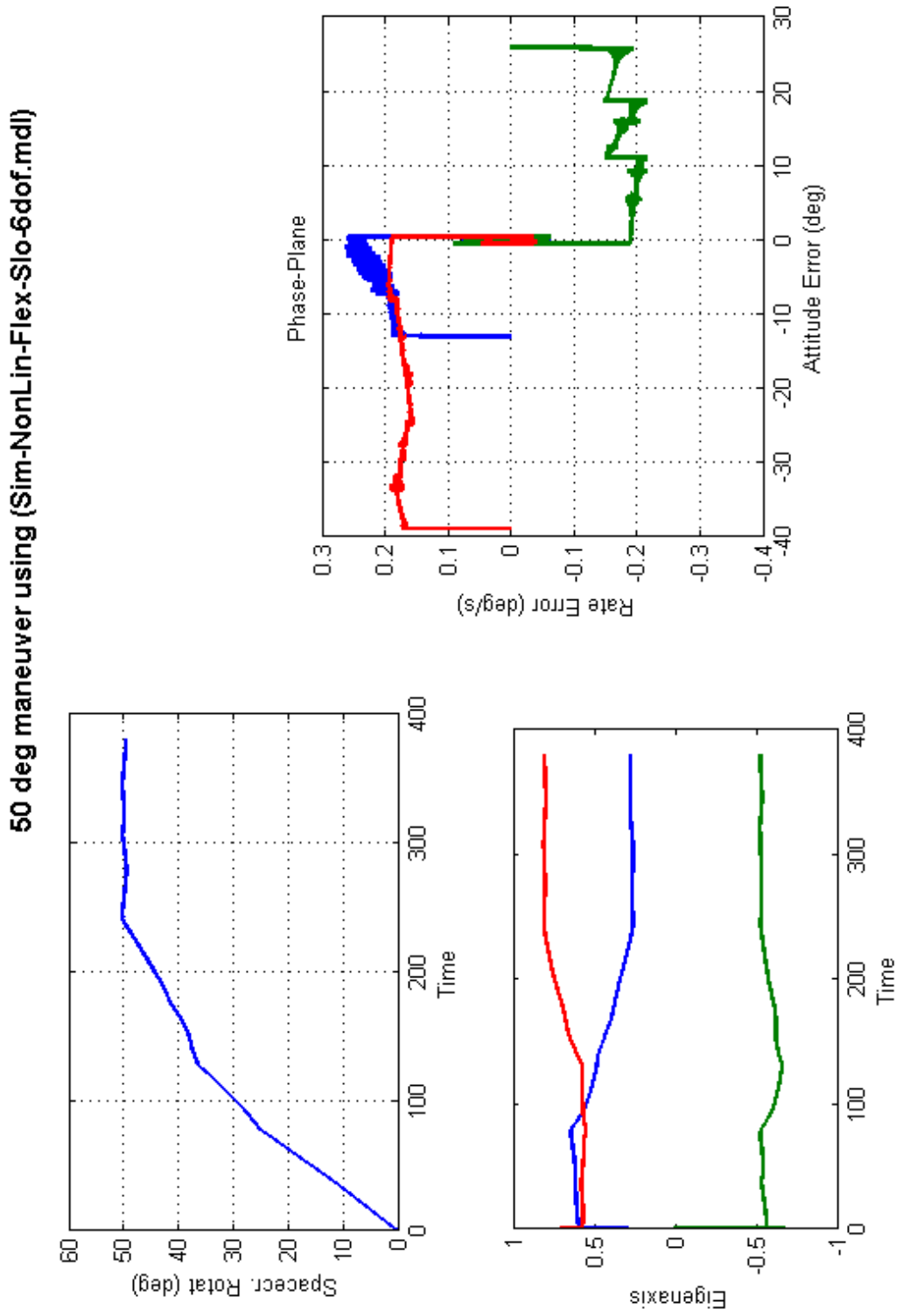
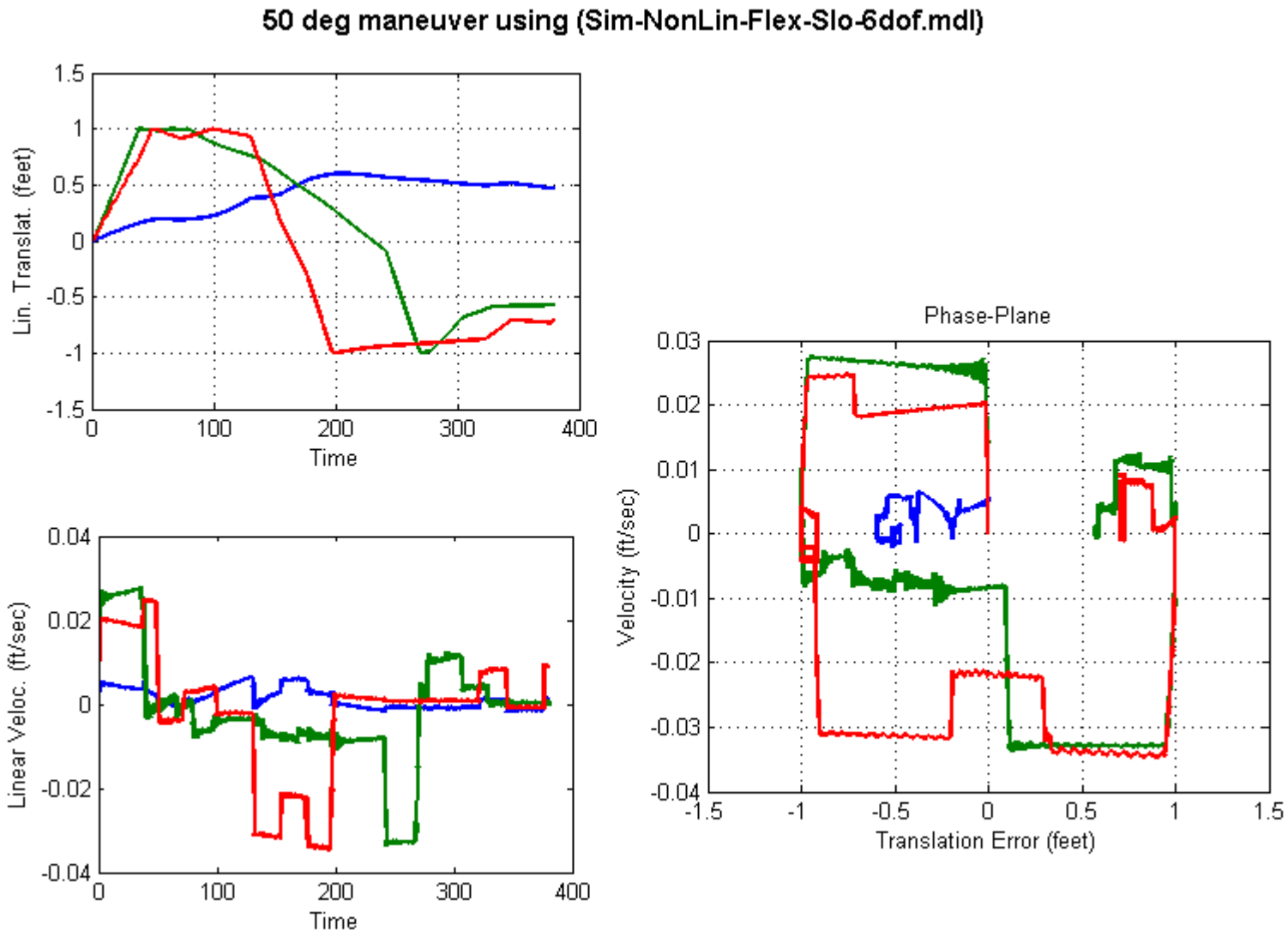
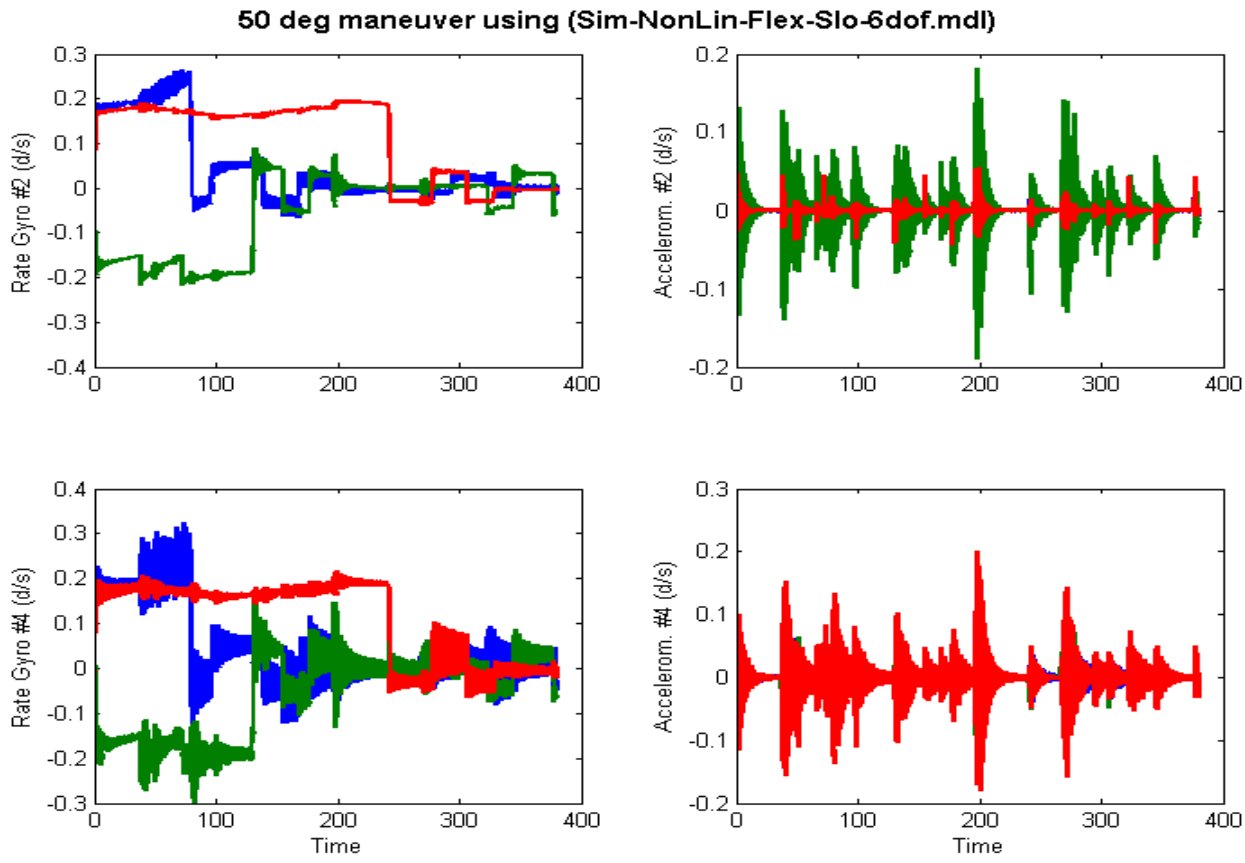
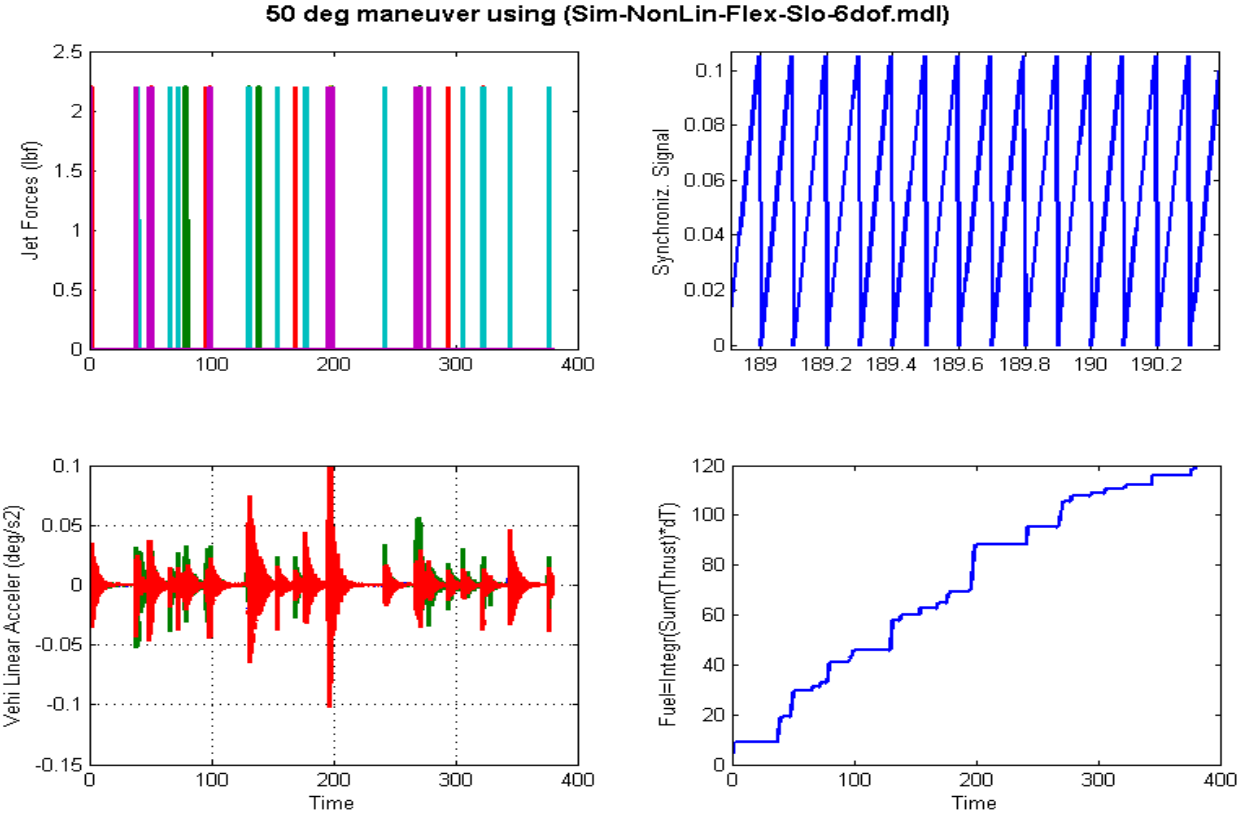


Figure 2.7.9 Spacecraft performs a 50 deg. attitude maneuver while maintaining a small translation of less than one foot


50 deg maneuver using (Sim-NonLin-Flex-Slo-6dof.mdl)
Lin. Translat. (feet)
Time
Linear Veloc. (ft/sec)
Time
Phase-Plane
Velocity (ft/sec)
Translation Error (feet)


50 deg maneuver using (Sim-NonLin-Flex-Slo-6dof.mdl)
Rate Gyro #2 (d/s)
Accelerom. #2 (d/s)
Rate Gyro #4 (d/s)
Accelerom. #4 (d/s)
Time
Time


50 deg maneuver using (Sim-NonLin-Flex-Slo-6dof.mdl)
Jet Forces (lbf)
Synchroniz. Signal
Vehi Linear Acceler (deg/s2)
Fuel=Integr(Sum(Thrust)*dT)
Time
Time

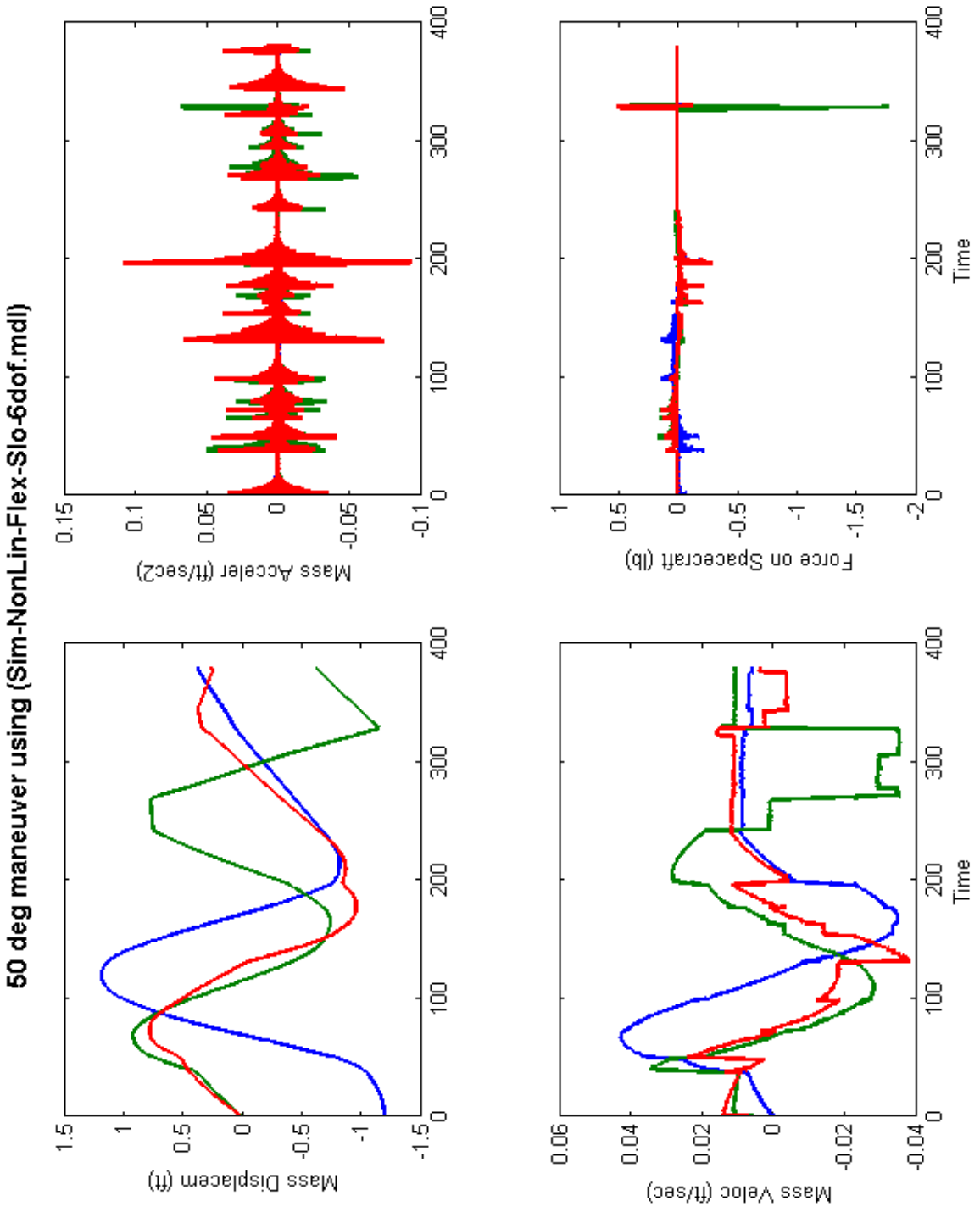
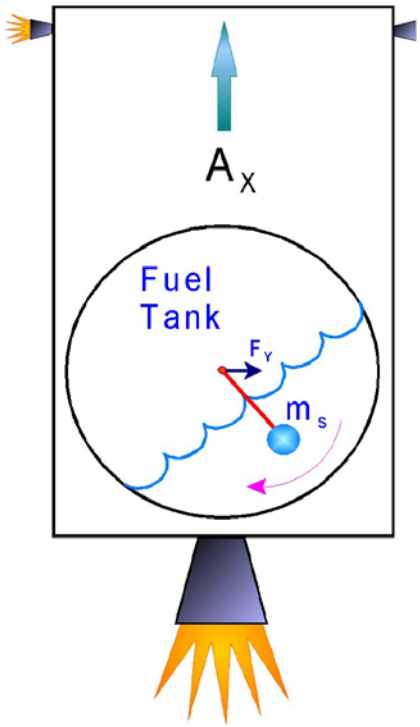


Figure 2.7.9 Slosh mass activity and reaction forces applied on the spacecraft

2.8 Reboost Analysis with the Main Engine Firing



During reboost the spacecraft fires the main engine to modify its altitude or change orbit. The ACS points the spacecraft in a proper orientation, the main engine ignites, and the RCS attempts to keep it at constant attitude or a slowly varying attitude during the orbital maneuver. We assume of course that the RCS can provide enough torque to correct any attitude errors that may occur due to misalignment of the thrust vector from the spacecraft CG. The translation control system is obviously turned off during this phase since we are constantly accelerating. The acceleration causes the fuel to accumulate towards the engine at the bottom of the tank, and any lateral disturbance causes sloshing which generate oscillatory disturbances on the spacecraft. The fuel dynamic behavior resembles that of a simple pendulum. When the desired orbit is reached the main engine is turned off and the ACS switches to a different mode of operation. In this section we will analyze a couple of accelerating reboost models: a linear model that includes a linear pendulum slosh model, and a non-linear model coupled with slosh and structural flexibility. The simulation files for this analysis are in folder: "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis**(i) Orbital Maneuvering-Reboost Phase". The file "*start.m*" initializes both simulation models.

2.8.1 Linear Reboost Simulation Model

The linear Simulink model for the reboost phase uses the state-space system that was generated using the Flixan "Flight Vehicle Modeling Program" in Section 1.2. Its title is "*Flexible Agile Spacecraft, Reboost Model (Z-Transf)*" and it was saved in file "*reboost_fvp_z.m*". In this case we are not using the complex zero-g slosh model wrapped around the spacecraft dynamics as in the previous zero-g case. This system is slightly different from the zero-g version used earlier because in this accelerating case the slosh dynamics simplifies to a linear pendulum resonance that can be included inside the vehicle state-space system. The pendulum frequency and other parameters are defined in the vehicle input data file. Also, in this case, the vehicle acceleration was set to 0.5 (ft/sec^2) in the x direction to capture the constant firing of the reboost engine. Otherwise, the program will not accept a slosh resonance when the vehicle acceleration is zero. Note, that in the vehicle data the slosh frequency is specified at 1g, (32.2 ft/sec^2) because it is usually known at 1g. The program adjusts the slosh frequency according to the total linear acceleration.

The simulation model for this linear case is in file "*Sim-Lin-Reboost-fvp.mdl*", shown in Figure (2.8.1). It uses the fuel optimal attitude control logic described earlier. There is no translation control during reboost. The flex spacecraft system with slosh is in Figure (2.8.2). The simulation results are shown in Figure (2.8.3). The spacecraft is commanded to maintain a constant zero attitude during reboost. The engine thrust is constant throughout the simulation at 110 (lb). There is

a repetitive RCS jet firing for counteracting the torque generated due to engine and CG misalignment. The RCS is holding the spacecraft attitude error below 0.5 (deg) defined by the dead-band.

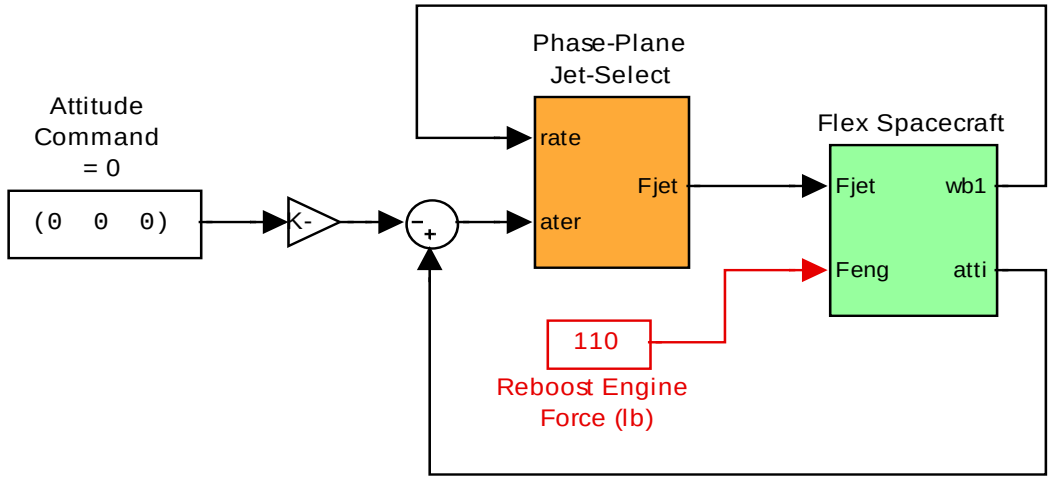
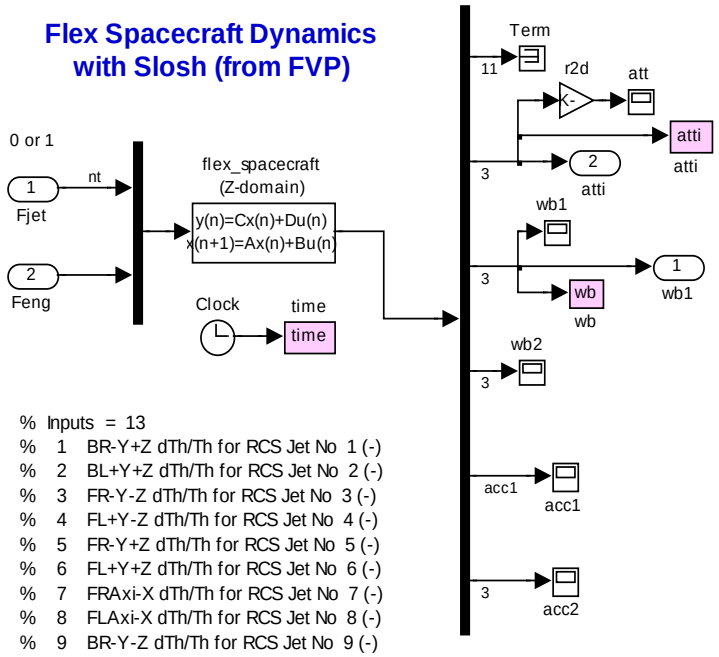


Figure 2.8.1 Linear Reboost Simulation Model "Sim_Lin_Reboost_fvp.mdl"



% Inputs = 13
% 1 BR-Y+Z dTh/Th for RCS Jet No 1 (-)
% 2 BL+Y+Z dTh/Th for RCS Jet No 2 (-)
% 3 FR-Y-Z dTh/Th for RCS Jet No 3 (-)
% 4 FL+Y-Z dTh/Th for RCS Jet No 4 (-)
% 5 FR-Y+Z dTh/Th for RCS Jet No 5 (-)
% 6 FL+Y+Z dTh/Th for RCS Jet No 6 (-)
% 7 FRAxi-X dTh/Th for RCS Jet No 7 (-)
% 8 FLAxi-X dTh/Th for RCS Jet No 8 (-)
% 9 BR-Y-Z dTh/Th for RCS Jet No 9 (-)
% 10 BL+Y-Z dTh/Th for RCS Jet No 10 (-)
% 11 BRAxi+X dTh/Th for RCS Jet No 11 (-)
% 12 BLAxi+X dTh/Th for RCS Jet No 12 (-)

% 13 Engine Force Direction: (1, 0, 0)

% Outputs = 26
% 1 Roll Attitude (phi-body) (radians)
% 2 Roll Rate (p-body) (rad/sec)
% 3 Pitch Attitude (thet-bdy) (radians)
% 4 Pitch Rate (q-body) (rad/sec)
% 5 Yaw Attitude (psi-body) (radians)
% 6 Yaw Rate (r-body) (rad/sec)
% 7 Angle of attack, alfa, (radians)
% 8 Angle of sideslip, beta, (radian)
% 9 Change in Altitude, delta-h, (feet)
% 10 Forw ard Acceleration (V-dot) (ft/sec)
% 11 Cross Range Velocity (Vcr) (ft/sec)

% 12 Gyro # 1, Roll Attitude (Body) (radians)
% 13 Gyro # 2, Pitch Attitude (Body) (radians)
% 14 Gyro # 3, Yaw Attitude (Body) (radians)

% 15 Rate-Gyro # 4, Roll Rate (Body) (rad/sec)
% 16 Rate-Gyro # 5, Pitch Rate (Body) (rad/sec)
% 17 Rate-Gyro # 6, Yaw Rate (Body) (rad/sec)

% 18 Rate-Gyro # 7, Roll Rate (Body) (rad/sec)
% 19 Rate-Gyro # 8, Pitch Rate (Body) (rad/sec)
% 20 Rate-Gyro # 9, Yaw Rate (Body) (rad/sec)

% 21 Accelerom # 1, (along X), (ft/sec^2) Translat. Acceler
% 22 Accelerom # 2, (along Y), (ft/sec^2) Translat. Acceler
% 23 Accelerom # 3, (along Z), (ft/sec^2) Translat. Acceler

% 24 Accelerom # 4, (along X), (ft/sec^2) Translat. Acceler
% 25 Accelerom # 5, (along Y), (ft/sec^2) Translat. Acceler
% 26 Accelerom # 6, (along Z), (ft/sec^2) Translat. Acceler

Figure 2.8.2 Spacecraft dynamic model with slosh created using the FVP

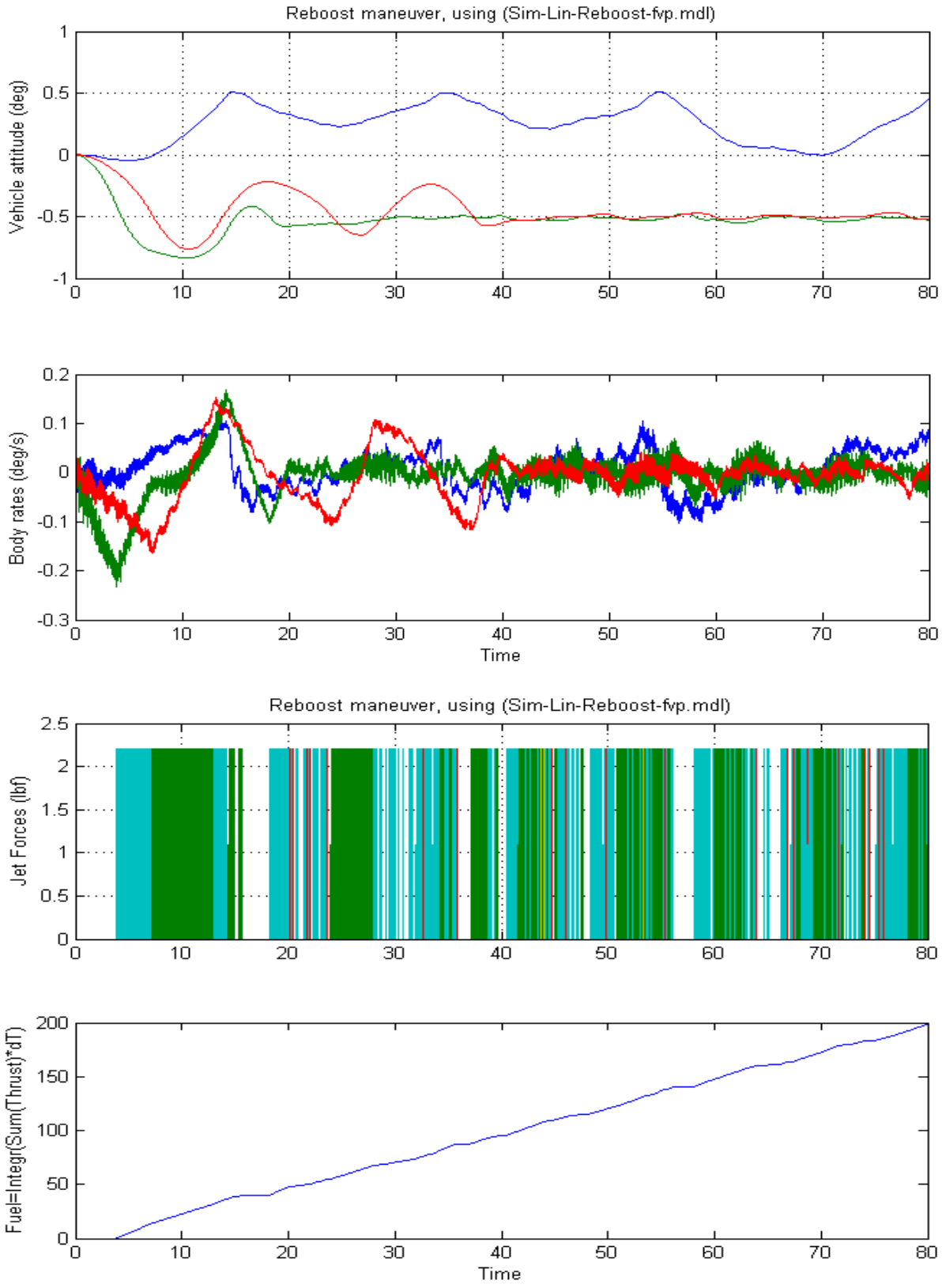


Figure 2.8.3 Simulation Results from the linear reboost system with slosh

2.8.2 Non-Linear Reboost Model

The non-linear Simulink model for the reboost phase is "*Sim_NonLin-Reboost_fem.mdl*", shown in Figure (2.8.4).

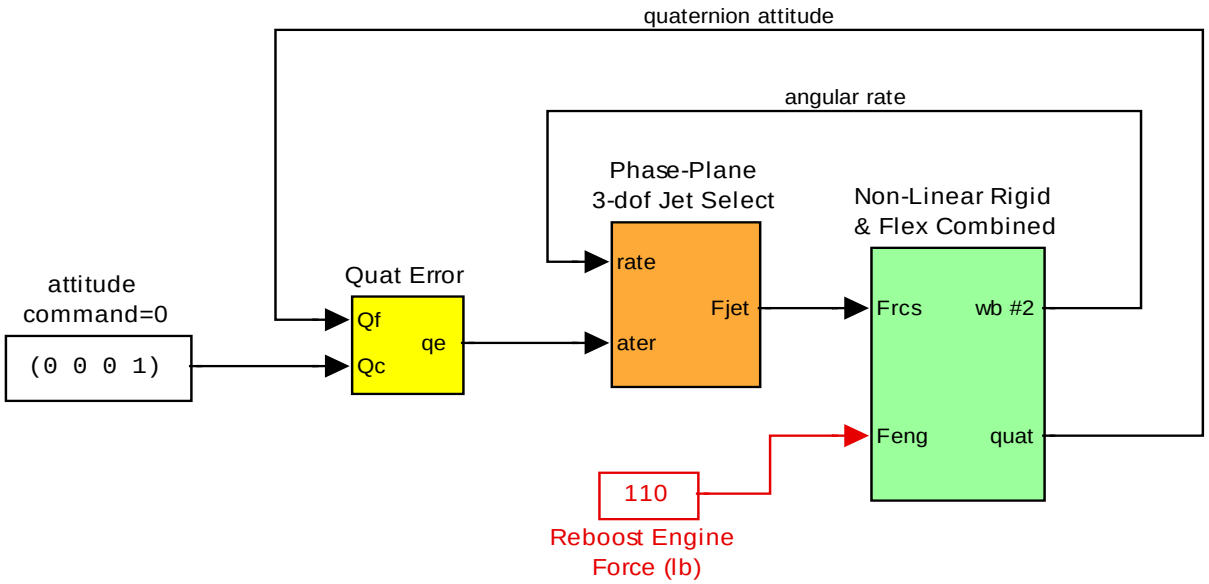


Figure 2.8.4 Orbital Reboost Simulation Model "Sim_NonLin_Reboost_fem.mdl"

The green spacecraft subsystem block is the same as the one used in Section 2.7. It includes the non-linear rigid-body function "*RB_Dynamics.m*", in parallel with the flex-only state-space model "*flex_only_fem_z.m*". The simulation parameters are initialized from file "start.m" and the file "pl.m" plots the simulation results, as soon as it ends. The spacecraft is commanded to maintain zero attitude during reboost, i.e. quaternion command = (0, 0, 0, 1). In the spacecraft dynamics block the non-linear slosh model closes the mechanical feedback loop by applying reaction forces at the tank as a function of the slosh mass motion relative to tank. The mass is initialized at an off-center position in the y direction to maximize the initial disturbance on the vehicle as it accelerates in the x direction. The main engine force is set to a fixed value of 110 (lbf) which provides a 0.45 (ft/sec^2) x-acceleration on the spacecraft. The simulation results are shown in Figure (2.8.5). The initial slosh mass displacement from the center causes an initial transient in the spacecraft attitude. The attitude error is significantly reduced as the mass converges (by means of a low damped oscillation) towards the main engine at the bottom of the tank (in the –x direction). Eventually, the vehicle maintains an average zero attitude within ±0.5 (deg) of error which is due to the dead-band. The constant acceleration causes the non-linear spring of the slosh model to deflect approximately 1.5 (inch), see figure (2.6.1). The RCS jets are steadily firing attempting to balance the disturbance torque generated by the engine thrust vector misalignment from the spacecraft CG. The plots also show two rate-gyro and two accelerometer signals in different spacecraft locations where the flexibility is different.

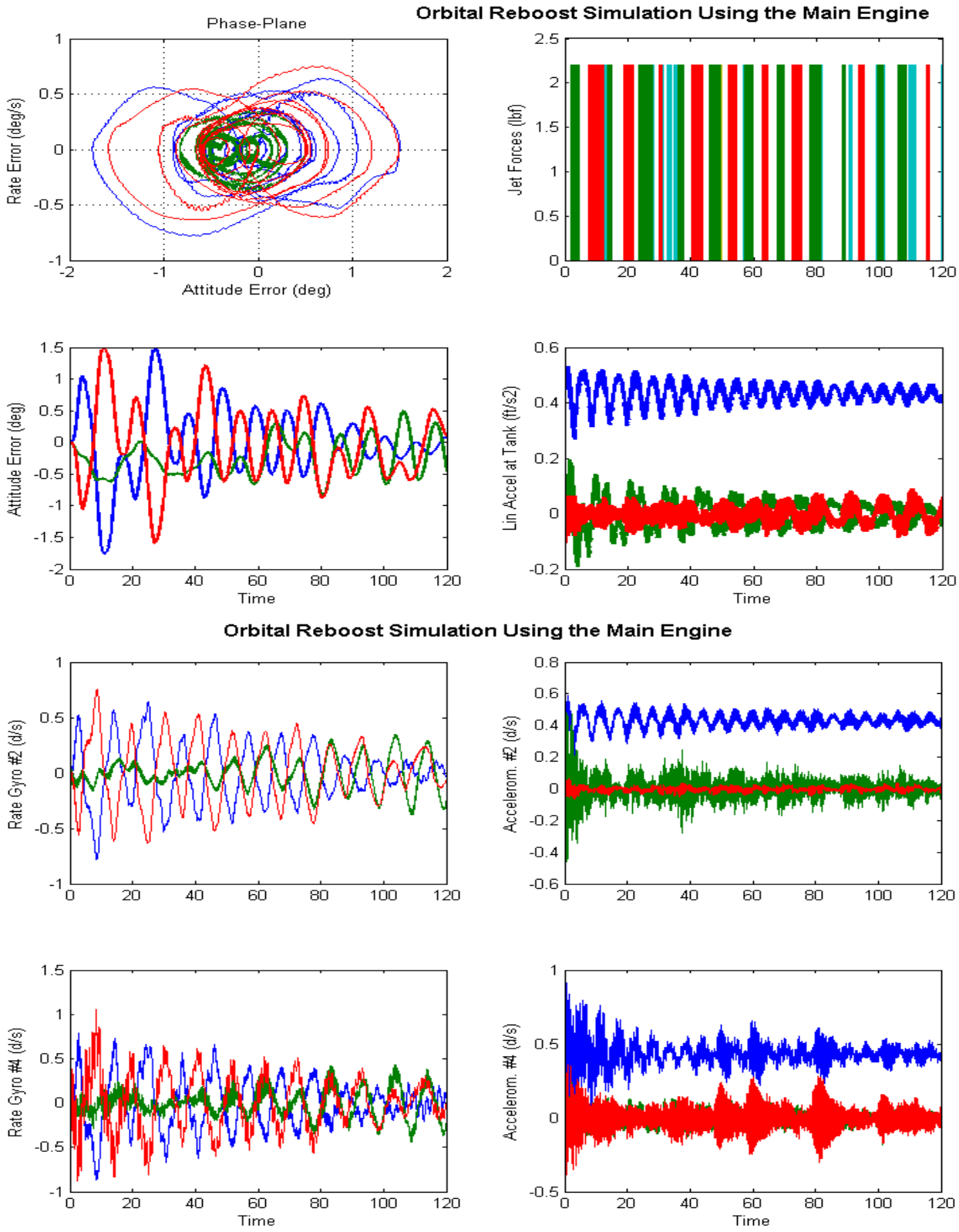


Figure 2.8.5 Non-Linear Reboost Simulation, Vehicle Response in Various Locations

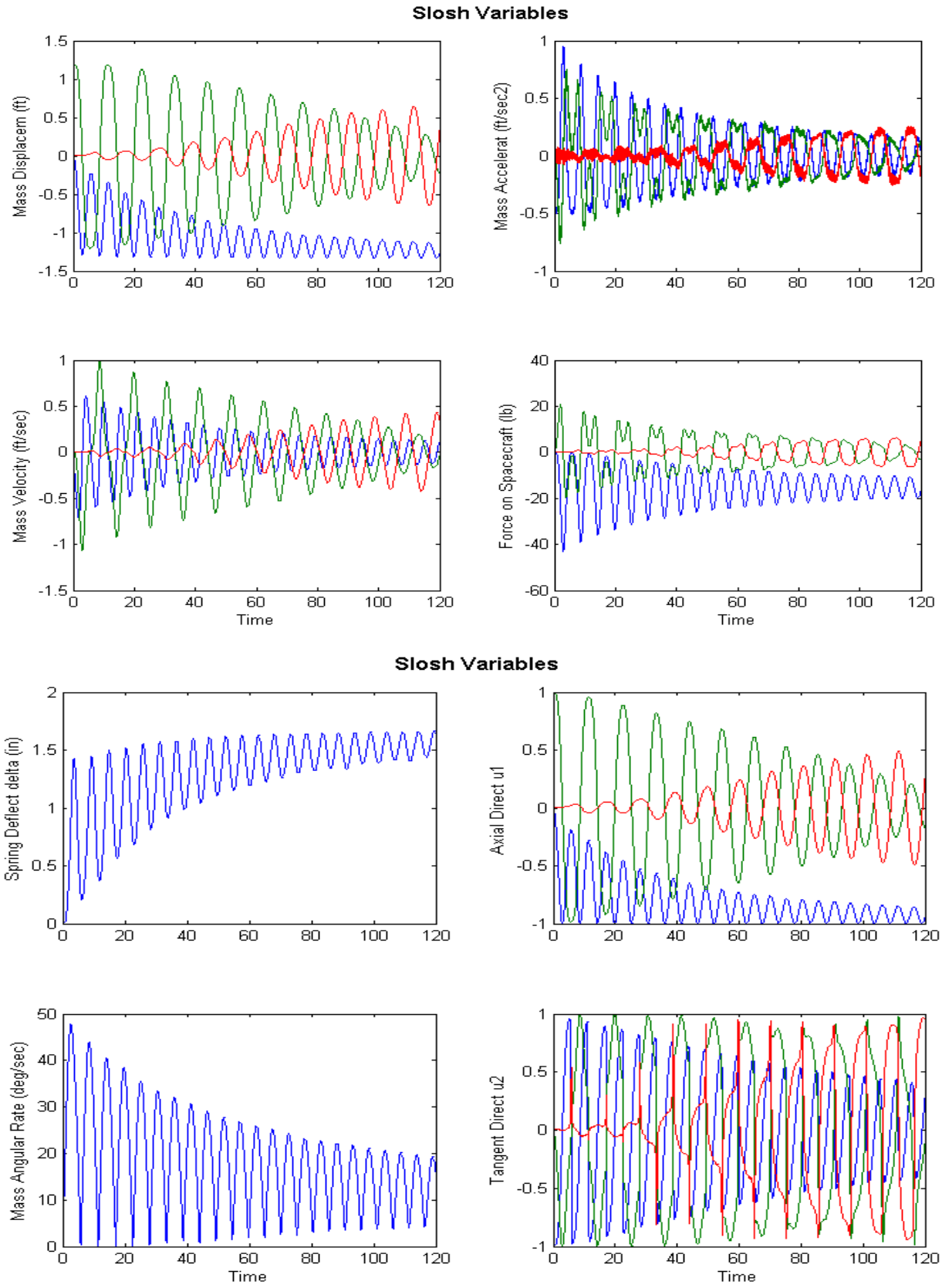


Figure 2.8.5 Non-Linear Reboost Simulation, Slosh Variables

2.8.3 Describing Function Analysis of the RCS during Reboost

The Describing Function (DF) is a very powerful frequency domain method for evaluating the stability margin of a non-linear control system, determining the existence of limit cycles, and also for designing filters to attenuate resonances and to shape the control system's frequency response in order to improve stability margin and to prevent limit-cycles. Limit cycles are sustained oscillations caused by non-linearities. We are not concerned with rigid-body limit-cycles because there is always going to be some degree of rigid-body limit-cycling. We are more concerned with limit-cycles which are caused due to structure flexibility excitation. It is very important, therefore, to achieve sufficient gain and phase margin in flex mode resonances in order to prevent structural limit-cycling because they are very undesirable. They use up a lot of fuel, corrupt measurements, spacecraft performance, and they cause structural damage due to metal fatigue. In the classical Describing Function methodology we separate a single-input-single-output (SISO) control systems in two parts: (a) the linear G(s) part which is a function of frequency (ω), and (b) the non-linear N(e) part which is a function of the error signal amplitude (e). Then we solve the feedback equation $G(j\omega)\ N(e) = 1$ by using Nichols or Nyquist diagrams. Intersections of the G(jω) locus with the -1/N(e) locus indicate the possibility of limit-cycles, but not every intersection defines a sustained oscillation. There are convergent limit-cycles and divergent limit-cycles, and they usually alternate. We are not concerned with divergent limit-cycles because they do not sustain an oscillation. We are only concerned with the convergent limit-cycles. The convergent limit-cycle amplitude and frequency can be obtained approximately from the loci intersections. The amplitude is obtained from the 1/N(e) locus, and the frequency is obtained from the G(jω) locus which are both co-plotted on a Nichols or a Nyquist plot.

It is not straightforward, however, how to apply the DF method in a typical RCS phase-plane system and requires some simplifications and assumptions. The classical DF method is applicable only to SISO systems, and it assumes that the DF of the non-linearity N(e) is a SISO, and a function of only amplitude but not frequency. In our situation, however, the control system structure is a little different and requires some assumptions to be made and block diagram manipulations in order to shape it in a form where the DF method can be applied. To start with, our phase-plane controller is unconventional because it has two inputs and one output. If we assume, however, that during limit-cycling the inputs are approximately sinusoidal, the two inputs: rate and attitude errors are related because the attitude error is the integral of the rate, hence, we can reduce the phase-plane inputs to only one input, the body rate and integrate the rate to get attitude. Any sustained limit-cycle generates a pattern in the phase-plane symmetric about the origin, producing, therefore, a periodic output torque with zero average, as shown in Figure (2.8.6).

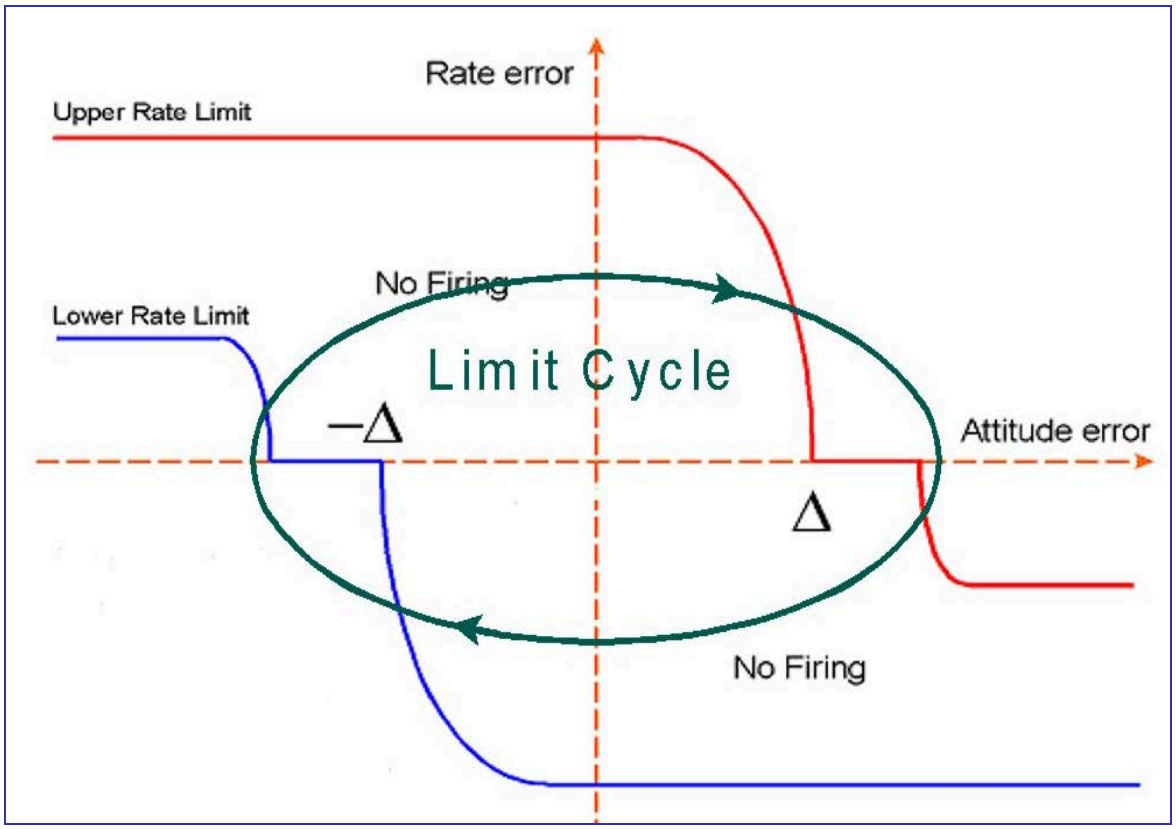


Figure 2.8.6 Sustained Limit-Cycle Trajectory in the Phase-Plane

The next problem to overcome is the fact that the output from the jet selection logic consists of multiple jet forces (F_{jet}). Remember, we need to separate the G(s) part from the N(e) part and, therefore, we must break the control loop at two points. One obvious point to separate the systems is at the spacecraft rate output which is the input to the non-linearity. We must also break the loop at the output of the non-linearity which is 12 jet forces. But it is not convenient to break the control loop at the jet force for stability analysis because the DF method requires a breaking point at a scalar, not a vector. One step closer to our goal is to transform the jet forces into torques (T) in the non-linear control system output and break the loop at the torque output. This transformation creates two separate (3x3) systems in a feedback loop, a non-linear part N(e) and a linear part G(s) that connect to each other by means of roll, pitch, and yaw rates and roll, pitch, and yaw torques, as shown in Figure (2.8.7). By cutting the loop at the torques it reduces the number of outputs to 3 instead of 12. The spacecraft plant inputs in this case must be torques (T). Now, each axis can be analyzed separately using the DF method, and that is not just roll, pitch, and yaw, but other skewed axes in between, because each direction uses a unique set of thrusters and excites the structure differently. The spacecraft torque is related to the jet forces by the following equation

$$T = V_t \, F_{jet}$$

Where:

$$V_t = \left(\underline{v}_1 \quad \underline{v}_2 \quad \ldots \quad \underline{v}_j \right) \; where: \underline{v}_j = \left(l_j \times u_j \right)$$

$$V_p = pseudo\_inverse\left(V_t \right)$$

Where:

l_j is the thruster (j) location relative to the spacecraft CG

u_j is the thrust direction for a thruster (j)

v_j is the torque on the spacecraft created by thruster (j)

But if the new plant input is torque, this torque it must be converted back into jet forces because the original plant model requires forces and, therefore, we use the pseudo-inverse of (V_t), V_{pinv}, to create pseudo forces to the plant input from torques, $F'_{jet} = V_{pinv}\, T$.

These forces will not be the same as the original forces F_{jet}, but they will produce the same amount of torque on the spacecraft. The biggest difference is that F'_{jet} does not excite the structure the same way as the original F_{jet}, but this is acceptable if we assume that the structure is sufficiently stiff between the jets, and that flexibility is mainly due to the appendages, such as, solar array, antennas, etc. It is, therefore, be acceptable to drive the plant input with F'_{jet} instead of the actual jet forces F_{jet}, as long as both force excitations create the same torque. Figure (2.8.7) shows the two (3x3) feedback interconnected systems: a non-linear (orange) block containing the controls and a linear spacecraft dynamics (green) block, interacting by means of torques and body rates. The attitude command is not shown because it does not affect stability.

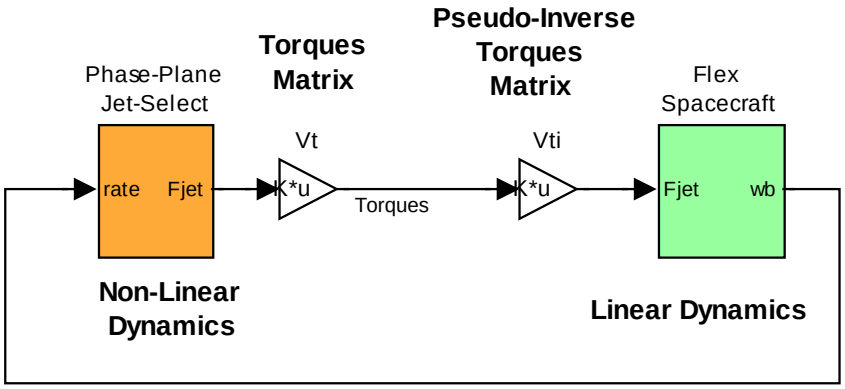


Figure 2.8.7 Modified Non-Linear System used for Describing-Function Analysis

The next step is to separate the above (3x3) systems to individual SISOs and analyze the dynamic motion and stability as if the spacecraft is excited and can move only in one direction at a time. The assumption is that since the system is strongly diagonally dominant by the selection of the jets, if it is stable in all individual SISO directions, it will also be globally stable when it is fully coupled. Let us assume, for example, that we are analyzing the pitch axis. We must first create a pitch SISO plant model by applying a scalar torque in pitch and measure the vehicle response only in pitch, as shown in Figure (2.8.8). Similarly, we can create a roll plant by changing the rotation input vector to (1 0 0), or a yaw plant (0 0 1), or a plan in any skewed direction, ex. (0.2 -0.4 0.5), defined by the rotational vector. We are ignoring, of course, the cross-coupling between axes in order to be able to use the DF method, but if the jets are properly selected the cross-axial coupling is negligible. So we can use the model in figure (2.8.8) to calculate the frequency responses of the plants G(s) in many different directions. The input is a scalar torque which is converted into vector torque in a specific direction and the output is a scalar body rate which is obtained from the spacecraft rate vector resolved in the same direction. This plant model connects with the phase-plane non-linearity of which we shall calculate its DF using Simulink.

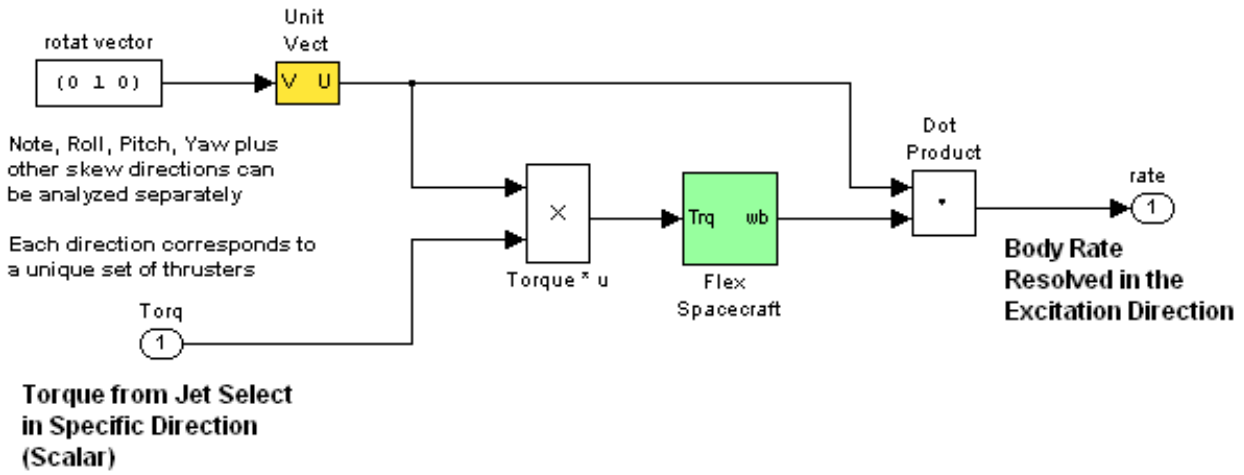


Figure 2.8.8 Plant Model Resolved in a Single Direction (pitch shown above)

Calculating the Describing Function of the Phase-Plane Non-Linearity

We can also apply the same approach to decouple the non-linear control system to behave like a SISO system in a specific rotational direction. Figure (2.8.9) shows the phase-plane non-linearity converted into a SISO Simulink block. The input is a scalar body rate feedback coming from the SISO vehicle model. It is converted to a vector (pitch rate in this example), and it is integrated to obtain attitude. We are assuming that the signal is sinusoidal since we are examining the possibility of limit-cycling. Attitude and rate errors drive the phase-plane (which is running slower, at 100 msec) and it generates the rate commands to the jet selection logic. The jet selection logic generates the jet forces, which turn-off sequentially during the 100 msec control cycle, and generate the control torque (T) after multiplying the acceleration matrix with the jet forces, $T = V_t F_{jet}$. The torque vector (T) is then resolved in a specific direction, pitch in this example, since the input rate was also in pitch. The model in Figure (2.8.9) is used to calculate the DF of the phase-plane and jet-selection logic together using Simulink. The direction of motion is defined by the rotation vector. Notice, that because of the integrator the DF in this case N(e,ω), is a function of both, error amplitude and frequency.

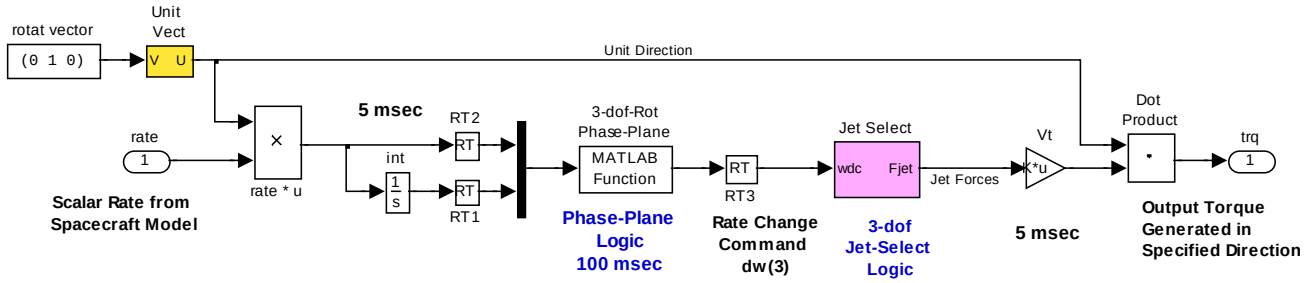


Figure 2.8.9 Non-Linear Control System Resolved in a Single Direction (pitch)

The DF of the control logic is too complicated to be derived analytically and we will use a Matlab program to calculate it from a Simulink model. The program that calculates the DF is "*DF_Calculate.m*" and it is saved in folder "… *\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\(i) Orbital Maneuvering-Reboost Phase\Describing Function Analysis\DF Calculate*". It executes a Simulink model "*DescFun_Sim.Mdl*" shown in Figure (2.8.10). The non-linearity in the orange block is the SISO control logic at a single axis, shown in figure (2.8.9).

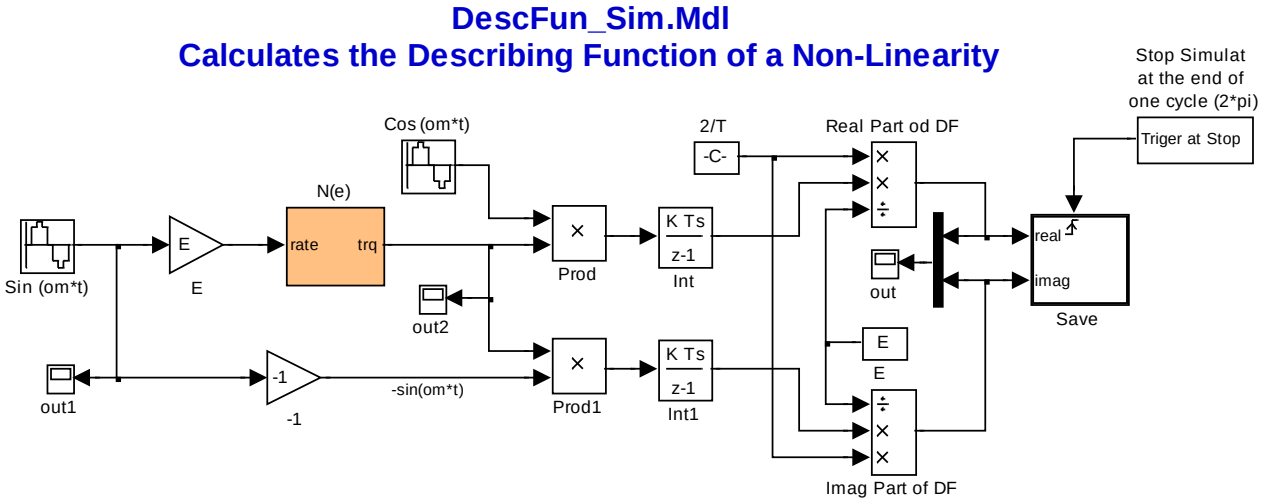


Figure 2.8.10 Simulink Model that Computes the Describing Function of the Non-Linearity N(e)

The DF program runs this simulation multiple times inside two nested loops, for a range of amplitudes and a range of frequencies. It calculates the Describing Function N(e,ω) as a function of input amplitude and input frequency and plots it in a 3-dimensional plot, as shown in Figure (2.8.11). The DF represents the gain of the non-linearity for a sinusoidal input at a specific amplitude and frequency, spacecraft torque over body rate in (ft-lb/rad/sec). The DF amplitude plots show that N(e,ω) is strongly dependent on the input amplitude (e), decreasing with amplitude. It decreases also with frequency but much less. Overall, the magnitude of the DF in yaw is about 10 (dB) higher than roll because it needs more torque in yaw having larger moment of inertia in yaw. The program also plots the (-1/N) magnitude versus phase that will be co-plotted with the Nichols of G(s) for stability analysis and limit cycles determination, see Figure (2.8.12). The (-1/N) loci are generated at specific frequencies of interest, starting at a low frequency 0.2 (rad/sec), the slosh frequency 0.72 (rad/sec), the roll appendage frequency 3.19 (rad/sec), and another appendage mode 5.84 (rad/sec) predominant in yaw.

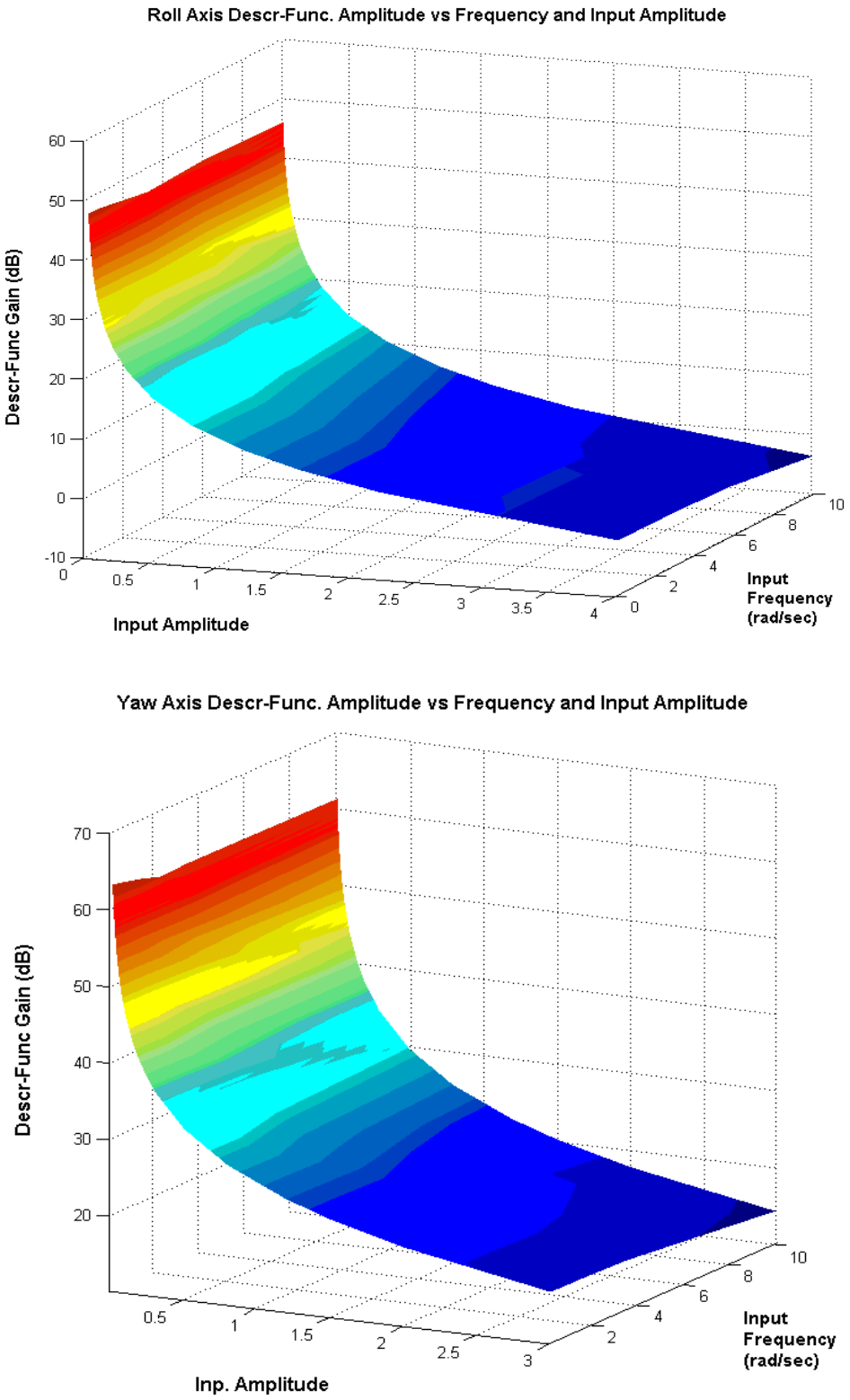


Figure 2.8.11 Describing Function of the Control System in Roll and Yaw as a Function of Spacecraft Input Rate (rad) and Input Frequency (rad/sec)

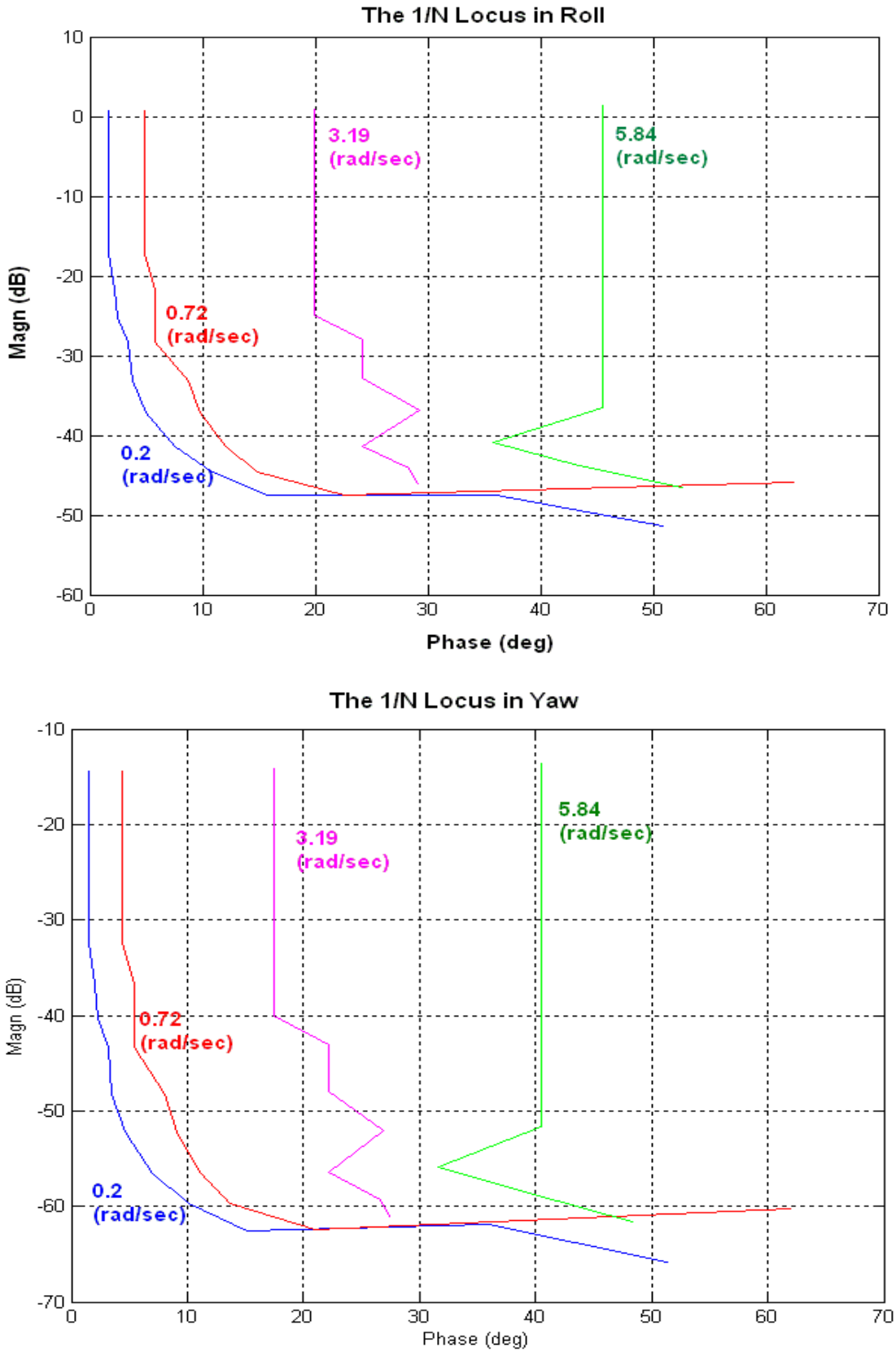


Figure 2.8.12 Inverse DF (-1/N) Loci Magnitude versus Phase for Stability Analysis

Plant Models

Just like in the time-domain simulation we have also created two different plant models for frequency domain stability analysis. The reason is to compare the efficiency of the two approaches in modeling the spacecraft dynamics, and to make sure that there are no modeling errors. The files for this frequency domain analysis are in folder "… *\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\ (i) Orbital Maneuvering-Reboost Phase\Describing Function Analysis*". The first plant model uses the spacecraft state-space system for the reboost phase "*reboost_fvp_s.m*", title "*Flexible Agile Spacecraft, Reboost Model*" that was created by the flight vehicle modeling program in Section 1.2. This model is very similar but not identical to the zero-g model that was used earlier. It includes the slosh mode of the fuel tank, and also has a non-zero axial acceleration Ax in the vehicle data that defines the slosh frequency. It is implemented as a continuous state-space model inside the Simulink model "*Open-Loop-FVP.mdl*", shown in Figure (2.8.4). It is forced to behave like a SISO system by directing the torque excitation in a specific rotational direction and also by resolving the motion in the same direction, as already discussed.

We also have a second spacecraft model that was created differently but it was also shaped in a similar structure, shown in Figure (2.8.13). This green spacecraft block consists of three separate subsystems which are combined together inside the Simulink model "*Open_Loop_FEM.mdl*", see Figure (2.8.14). The rigid-body dynamics is implemented inside the Matlab function "*RB_Dynamics.m*". The flex dynamics, consisting only of flex modes, is loaded as a continuous state-space system from file "*flex_only_fem_s.m*", which was created in Section 2.1. The third subsystem inside the Simulink model is a linearized version of the slosh pendulum model. This is implemented inside the Matlab function "*Slosh_Lin.m*". It creates forces at the tank as a function of vehicle acceleration, similar to the non-linear slosh model discussed earlier. The yellow "Unit Vector" block guarantees that the rotational direction entered is converted into a unit vector.

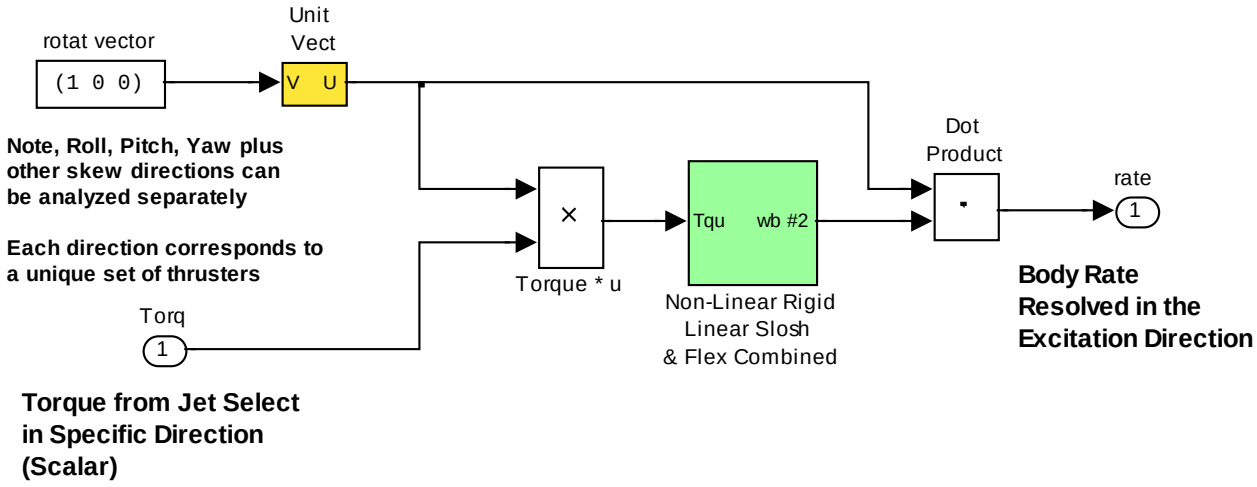


Figure 2.8.13 Simulink Model "Open_Loop_FEM.mdl" for Frequency Domain Analysis using the DF

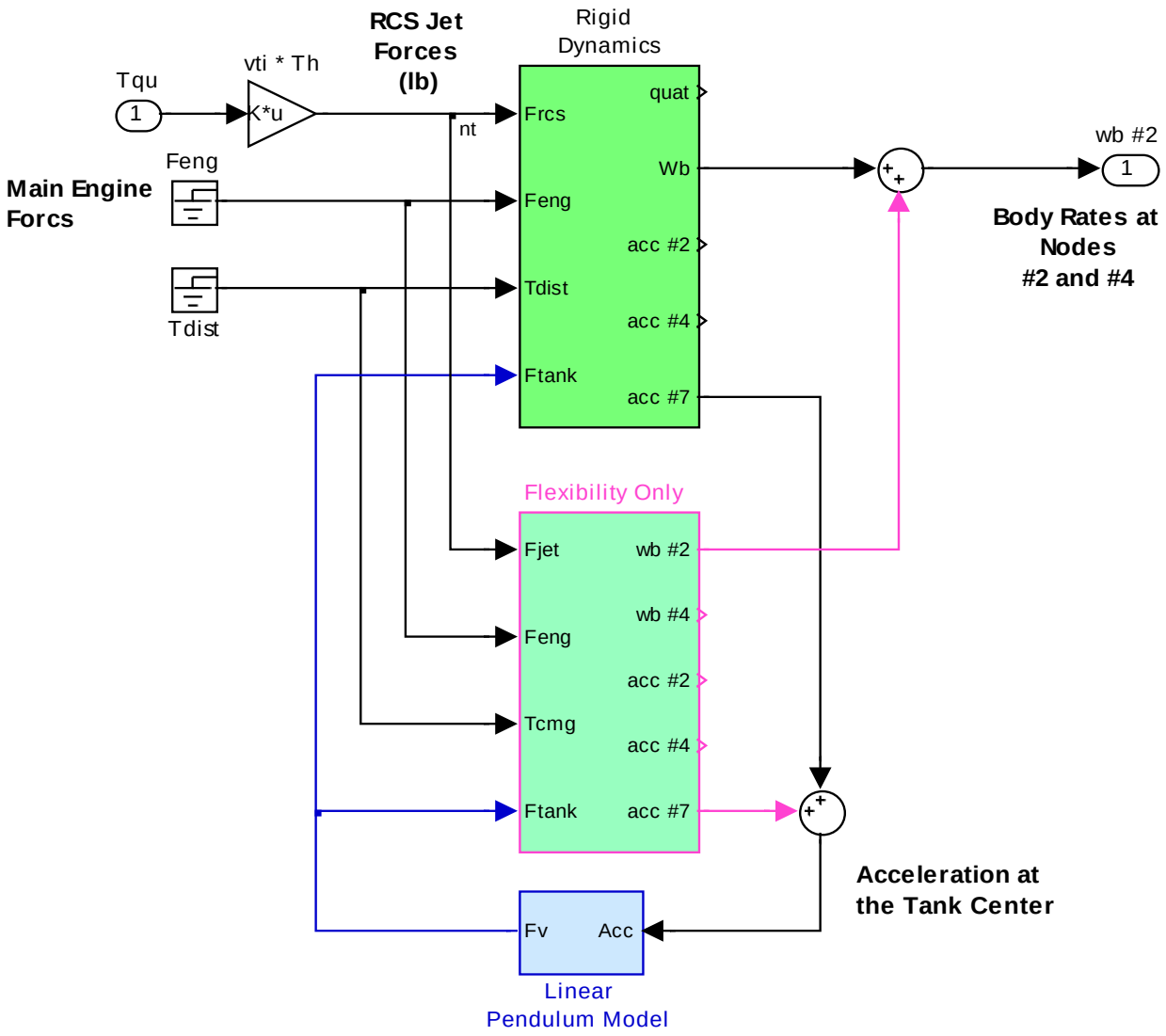


Figure 2.8.14 Spacecraft Dynamics Block inside "Open_Loop_FEM.mdl"

The file "run_freq.m" loads the system parameters and calculates frequency responses in roll and yaw from the two spacecraft models. The results from the two systems are co-plotted in Figure (2.8.15) and they are identical in all directions.

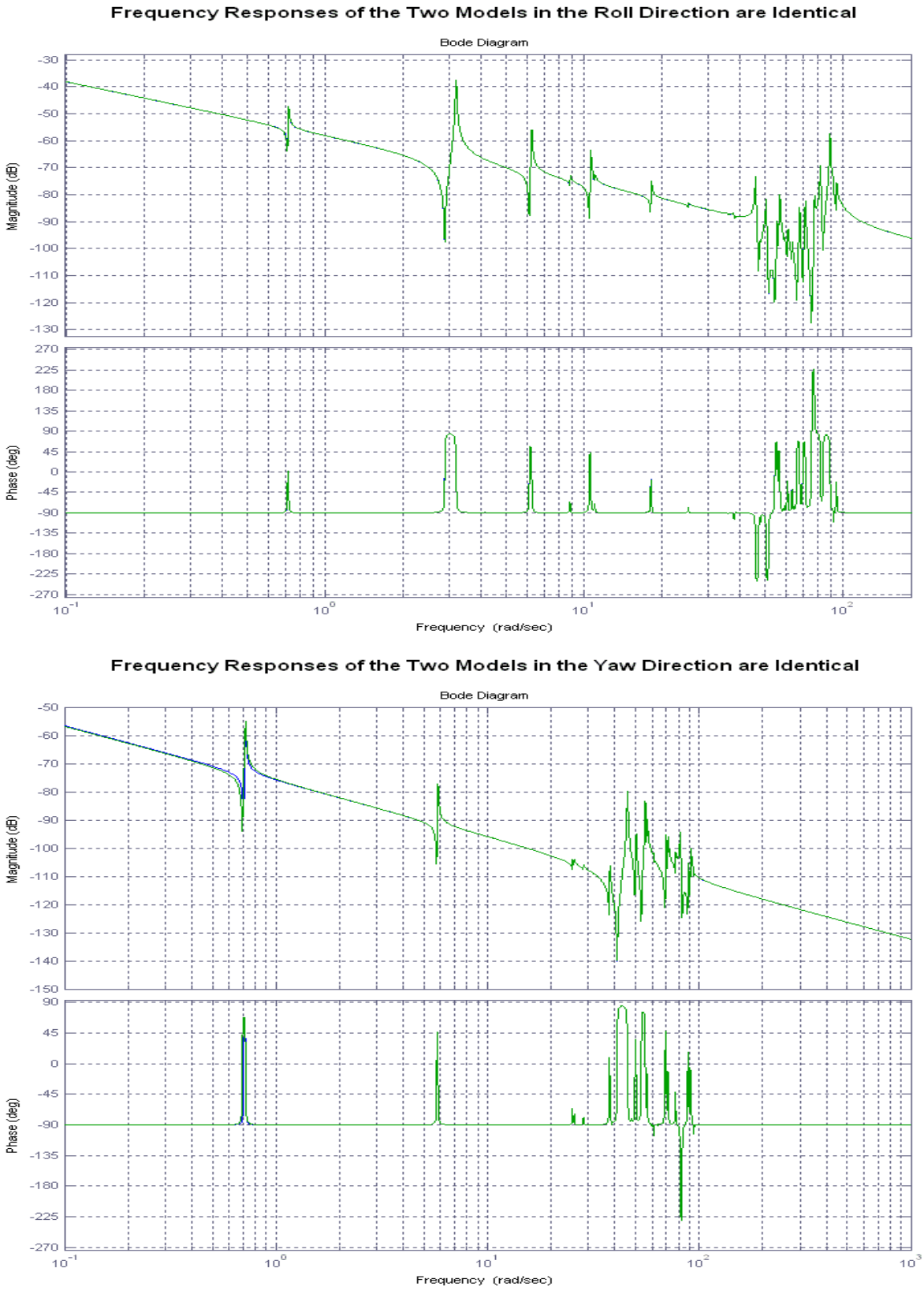
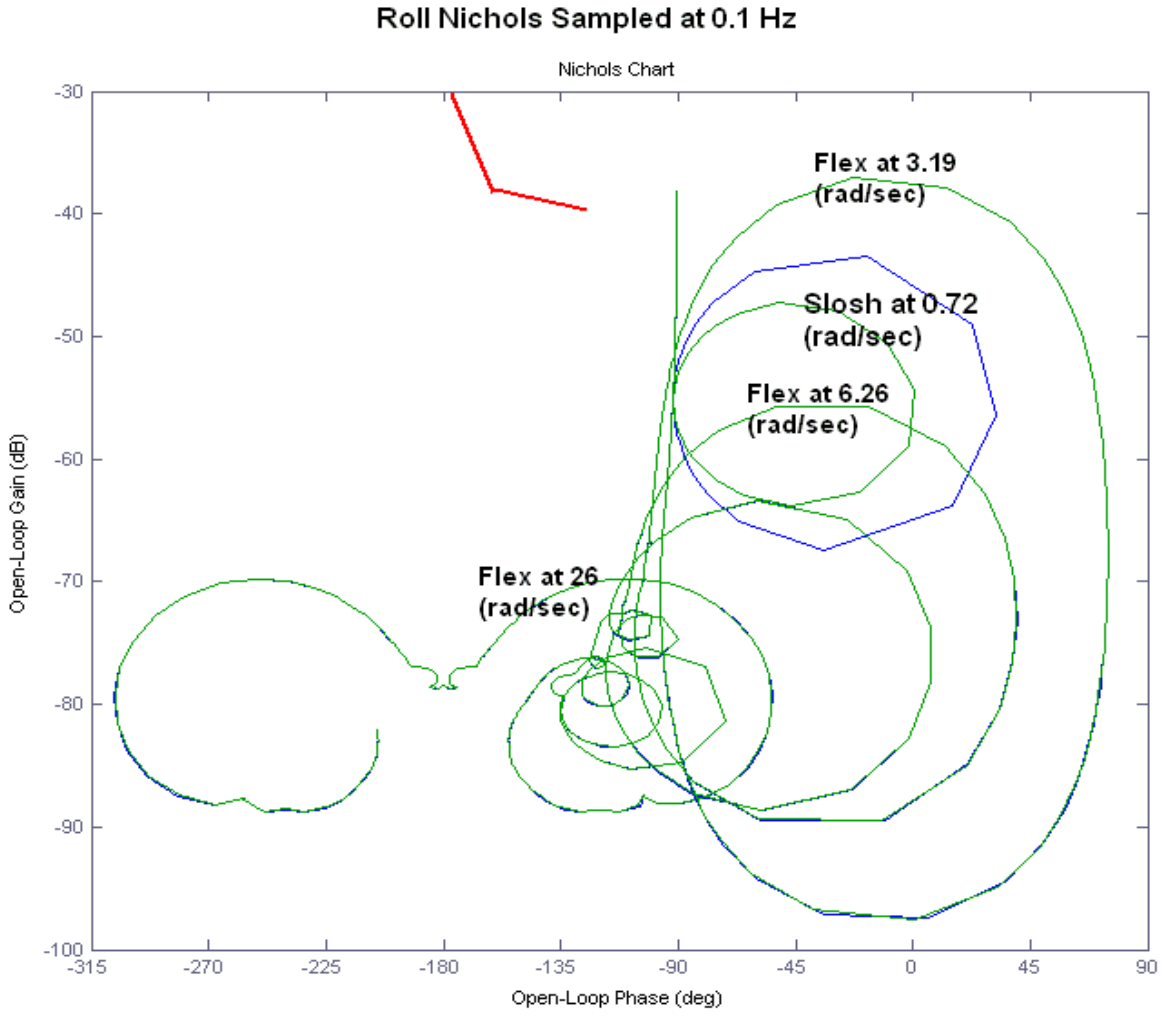
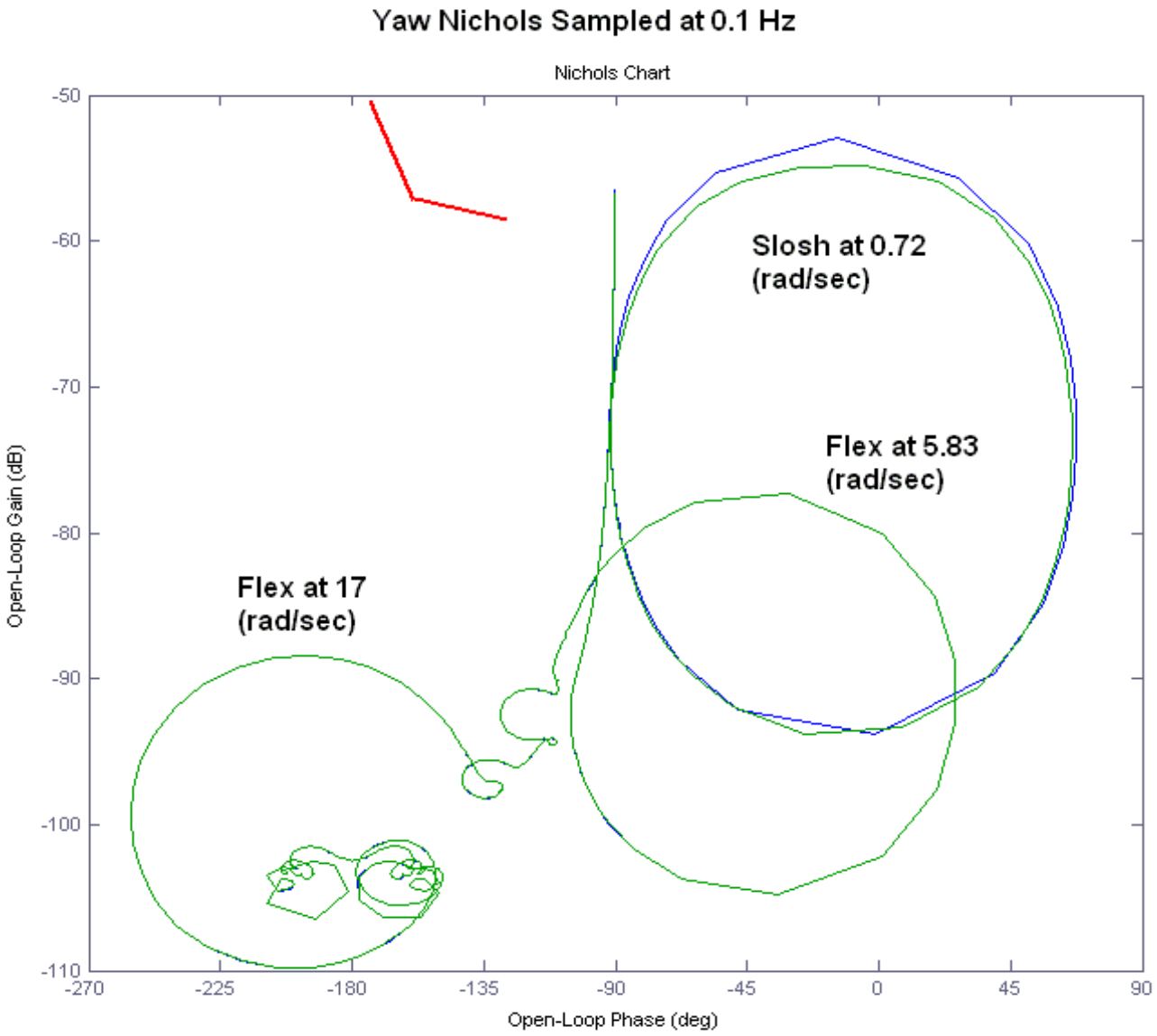


Figure 2.8.15 Frequency Responses from the Two Systems are Identical in all Directions

Stability Analysis Using the DF

The sampling frequency of the control loop is at 10 Hz and, therefore, in the DF analysis the plant should be sampled at 10 Hz. In the DF analysis we co-plot the Nichols of the G(s) with the -1/N(e) locus and hope that there are no intersections. The DF in this case is frequency dependent and it would require a 3-d illustration but we can take advantage that the -1/N(e) does not vary much with frequency. We can plot only a few of the loci, $-1/N(e,\omega_1)$, $-1/N(e,\omega_2)$, $-1/N(e,\omega_3)$, where ω_i correspond to some of the big resonances, that stick out, such as slosh and a couple of appendage modes below 5 Hz and overlay them with the G(s) of the linear plant. This analysis must be repeated for at least 3 directions separately, roll, pitch, yaw, plus a few additional skewed directions, but we are only showing a couple cases. The file "run_freq" discretizes the two Simulink models "Open_Loop_FEM.mdl" and "Open_Loop_FVP.mdl" and calculates the Nichols plots shown in Figure (2.8.15) for roll and yaw. The Describing Function overlay was done manually from the plots in Figure (2.8.12). There is a slight difference in the size of the slosh resonance between the models, appearing in the roll direction. The results clearly indicate that this system is not threatened by limit cycles in neither direction. In fact, all resonances are well behaved in phase, exhibiting min-phase type behavior.




Yaw Nichols Sampled at 0.1 Hz
Nichols Chart
Slosh at 0.72
(rad/sec)
Flex at 5.83
(rad/sec)
Flex at 17
(rad/sec)
Open-Loop Gain (dB)
Open-Loop Phase (deg)
-50
-60
-70
-80
-90
-100
-110
-270
-225
-180
-135
-90
-45
0
45
90

3. Spacecraft Controlled by an Array of Single Gimbal Control Moment Gyros

This section provides a detailed tutorial on designing spacecraft Attitude Control Systems using Single-Gimbal Control Moment Gyro devices. We will use the same spacecraft model that was described in the previous section using RCS and design an alternate ACS that uses SG-CMGs. The ACS consists of a cluster of four SG-CMGs mounted together on a solid structure near the center of the spacecraft. They are controlled by a non-linear Max Energy control logic that attempts to use the maximum control torque and momentum capability of the CMG devices to improve (in comparison with linear controls) the spacecraft ability to maneuver. The control law also uses a steering logic that prevents CMG singularities. Singularities or “gimbal locks” occur when the SGCMG cluster cannot provide torque in a required direction. In the following sections we will discuss the SGCMG dynamics, the equations of motion of a spacecraft with SG-CMGs, design the control and steering laws, and describe the simulation models. We start with simple rigid-body models that exclude the CMG gimbal dynamics and the flexibility of the structure, and we gradually upgrade the models with more details. We finally analyze a low cost configuration that uses a combination of one reaction wheel and two CMGs. The flex spacecraft models are created using two separate Flixan modeling programs for comparison.

3.1 CMG Array Geometry

A single gimbal control moment gyroscope (SGCMG) is shown in Figure 3.1. It consists of a spinning rotor that is mounted on gimbal perpendicular to the rotor axis. The rotor spin rate is maintained at a constant speed by a small motor that produces a constant angular momentum (h_{cmg}). The momentum direction can be rotated by a stronger motor which is mounted at the gimbal. The gimbal motor controls the gimbaling rate, and hence the output torque. By commanding the gimbal to rotate (by means of a servo system that controls the gimbal motor), high precession torques are generated by changing the orientation of the angular momentum vector. The reaction torque on the spacecraft (T) is equal and opposite to the rate of change in momentum vector $\dot{h}$, which is orthogonal to the momentum vector and the gimbaling vector according to the right hand rule. At any instant it is a function of the gimbal position. In fact the torque magnitude is equal to the CMG momentum multiplied by the gimbal rate. A CMG generates much greater torques than a reaction wheel and for this reason it is very attractive in high torque and fast maneuvering spacecraft applications. Notice also, that the SG-CMGs generate the high precession torques without requiring high power.

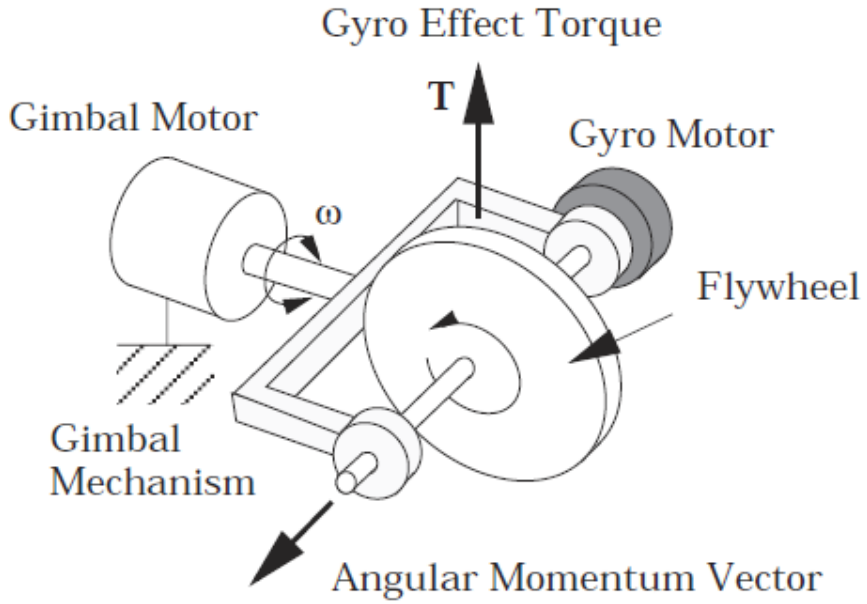
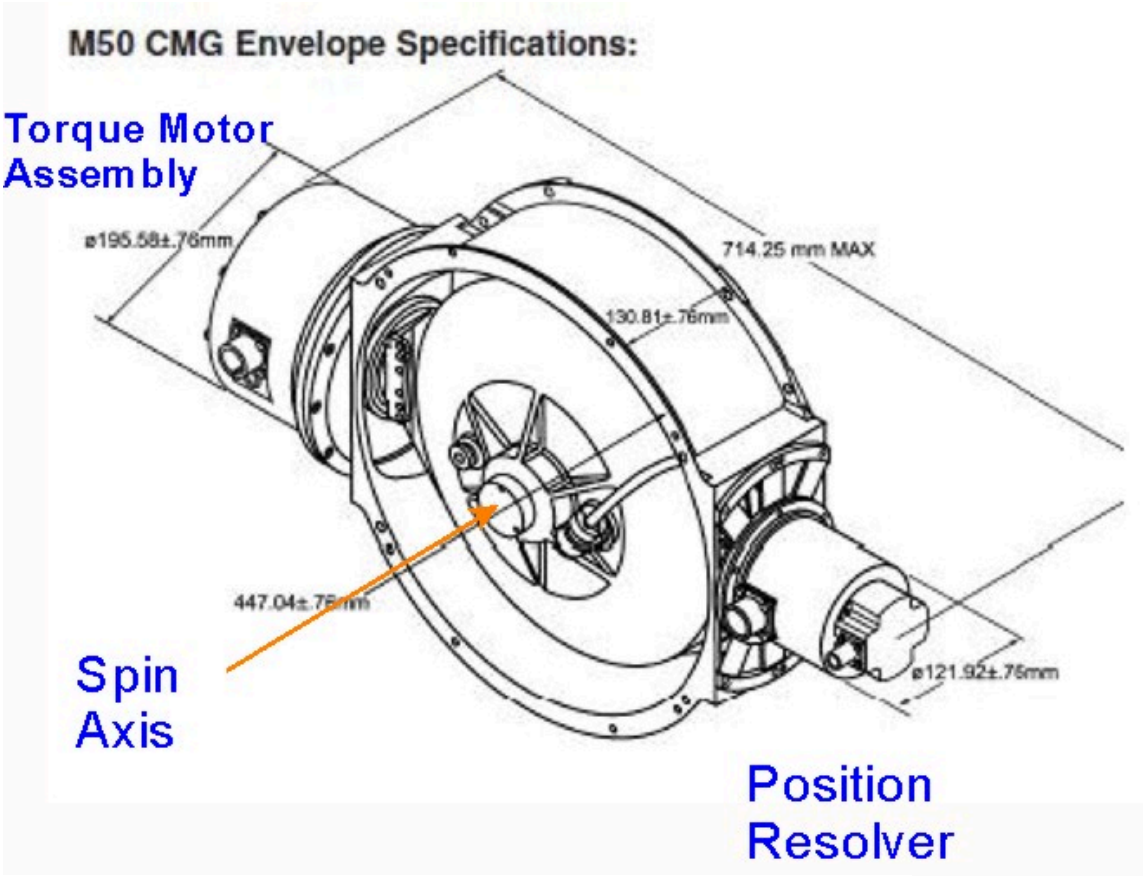


Figure 3.1 Single-Gimbal Control Moment Gyro

Another attractive feature of CMGs compared with reaction wheels is that the rotor in a CMG spins at a constant rate which places the vibrations at known frequencies while in a RW, the rotor speed changes, thus, exciting the spacecraft structure in multiple frequencies which may not be desirable in precision applications. CMGs, however, are complex systems, expensive, and require complex controls with singularity avoidance algorithms. Figure 3.2 shows a picture of a SG-CMG. It consists of a rotor that spins at a constant high rate about an axis that can be rotated. There is also a torque motor assembly that rotates the rotor about a gimbal axis that is fixed relative to the spacecraft, and a position sensor that measures the gimbal rotation relative to the spacecraft.



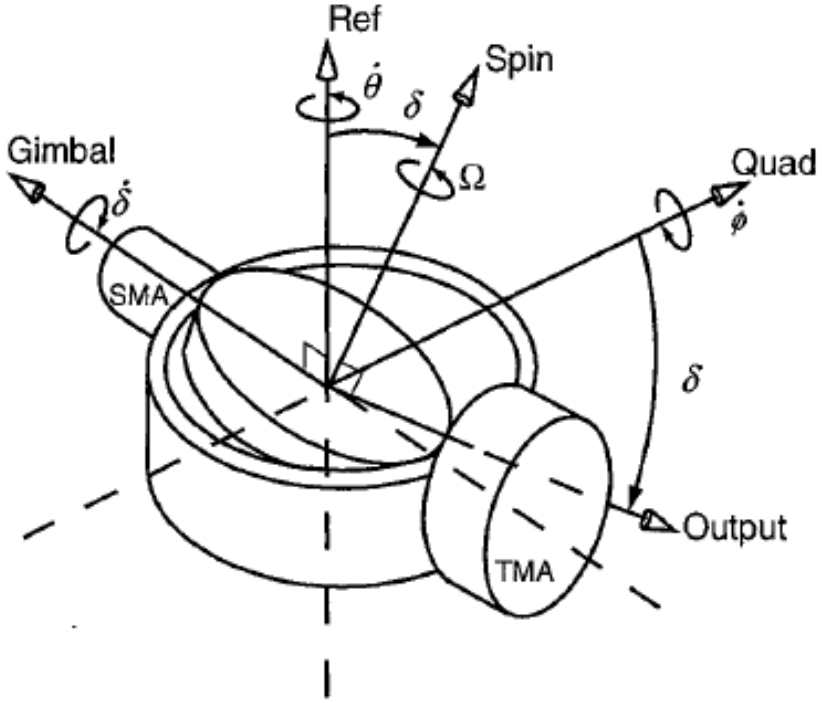


Figure 3.2 Single Gimbal Control Moment Gyro

The momentum of one CMG about the Gimbal, Output, and Spin axes can be calculated as a function of the spinning rotor momentum and the spacecraft rotation rate in CMG axes.

$$\begin{pmatrix} h_g \\ h_o \\ h_s \end{pmatrix} = \begin{pmatrix} J_g \dot{\delta}_i \\ J_o\left(\dot{\phi}\cos\delta - \dot{\theta}\sin\delta\right) \\ h_0 + J_s\left(\dot{\theta}\cos\delta + \dot{\phi}\sin\delta\right) \end{pmatrix} \qquad (3.1)$$

Figure 2.3 defines the orientation of a Single Gimbal CMG vectors with respect to the spacecraft axes. The rate of change of momentum which is the moment generated by a SGCMG in the Gimbal, Output, and Spin axes respectively as a result of gimbaling and base motion are:

$$\begin{bmatrix} M_G \\ M_O \\ M_S \end{bmatrix} = \begin{bmatrix} T_{gi} \\ J_o\left(\ddot{\phi}\cos\delta - \dot{\phi}\dot{\delta}_i\sin\delta - \dot{\theta}\dot{\delta}_i\cos\delta - \ddot{\theta}\sin\delta\right) + h_0\dot{\delta}_i + \dot{\delta}_i\left(J_s - J_g\right)\left(\dot{\theta}\cos\delta + \dot{\phi}\sin\delta\right) \\ J_s\left(\dot{\Omega} + \ddot{\theta}\cos\delta + \ddot{\phi}\sin\delta + \dot{\phi}\dot{\delta}_i\cos\delta - \dot{\theta}\dot{\delta}_i\sin\delta\right) + \dot{\delta}_i\left(J_g - J_o\right)\left(\dot{\phi}\cos\delta - \dot{\theta}\sin\delta\right) \end{bmatrix} \qquad (3.2)$$

The reaction torque on the spacecraft is minus [M_G, M_O, M_S]

Where:

J_g is the CMG inertia about its gimbal axis
J_o is the CMG inertia about its output axis
J_s is the CMG inertia about its spin axis
T_{gi} is the torque applied by the torque motor at the gimbal
$\ddot{\delta}_i$ is the gimbal inertial angular acceleration including spacecraft
δ is the CMG gimbal rotation about the m axis with respect to spacecraft r axis
h_0 is the constant CMG momentum about its spin axis ($I_s\Omega$)
$\dot{\Omega}$ is the rotor spin acceleration
$\dot{\theta}$ is the vehicle rate in the CMG r axis
$\dot{\phi}$ is the vehicle rate in the CMG q axis

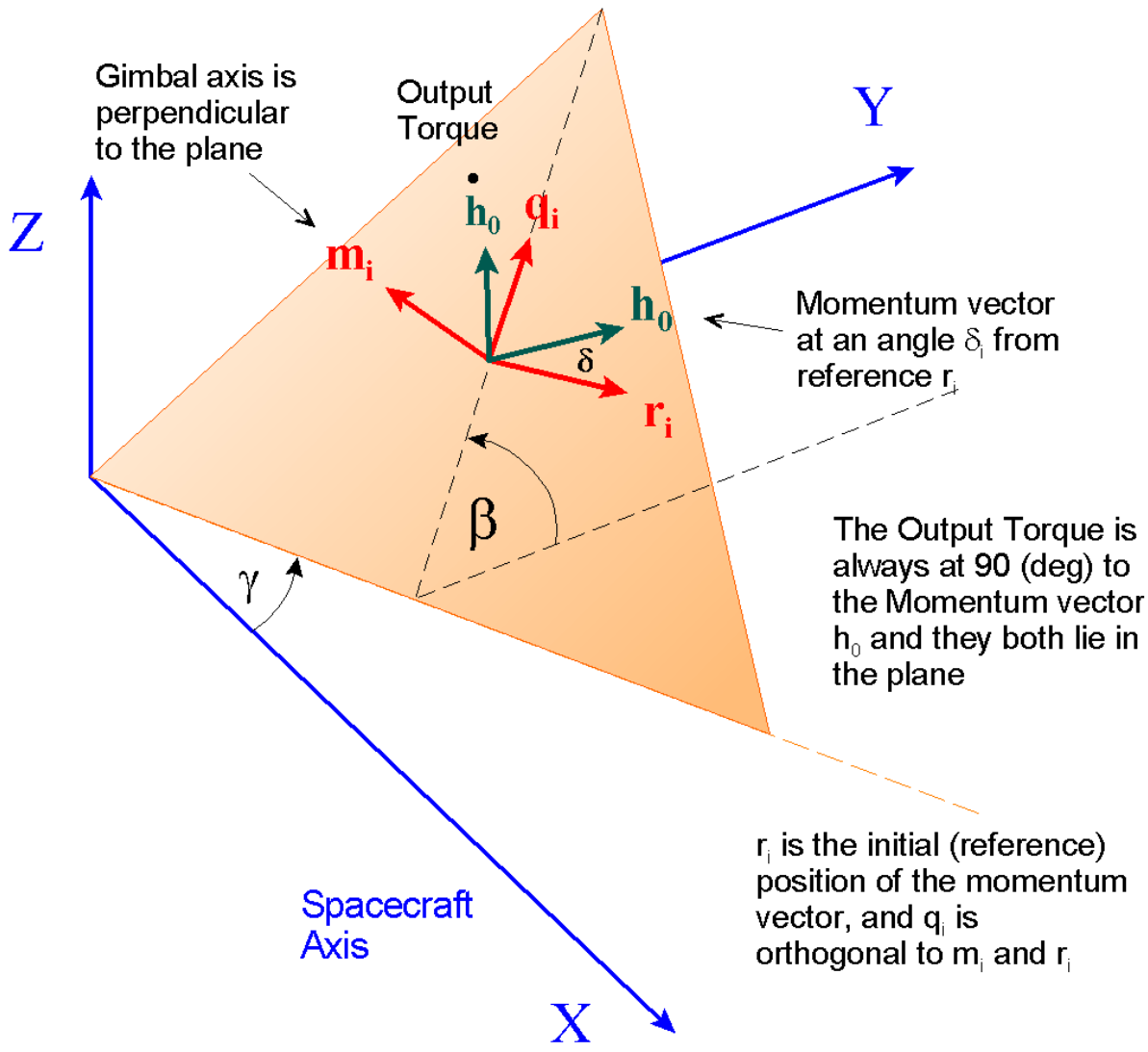


Figure 3.3 Orientation of a CMG in Spacecraft Coordinates

The rotation rates of the CMG axes can be related to the spacecraft body rate. If $\underline{\omega}$ is the spacecraft body rate vector (ω_X , ω_Y , ω_Z), the following relationships resolve the spacecraft rates about the CMG axes: (r, q, m). $\dot{\theta}$ *and* $\dot{\phi}$ are the spacecraft rates resolved about the CMG reference and quad axes. The gimbal rate $\dot{\delta}_i$ in addition to the gimbal rate relative to spacecraft $\dot{\delta}$ it includes also the spacecraft rate about the CMG gimbal.

$$\begin{aligned}
\dot{\theta} &= \omega_X \cos\gamma + \omega_Y \sin\gamma \\
\dot{\phi} &= -\omega_X \cos\beta \sin\gamma + \omega_Y \cos\beta \cos\gamma + \omega_Z \sin\beta \\
\dot{\delta}_i &= \dot{\delta} + \omega_X \sin\beta \sin\gamma - \omega_Y \sin\beta \cos\gamma + \omega_Z \cos\beta
\end{aligned} \qquad (3.3)$$

The following projection matrix (P) transforms the CMG torques from CMG axis to spacecraft axis.

$$\begin{pmatrix} M_X \\ M_Y \\ M_Z \end{pmatrix} = \begin{bmatrix} \sin\beta\sin\gamma & -\sin\delta\cos\gamma - \cos\delta\cos\beta\sin\gamma & \cos\delta\cos\gamma - \sin\delta\cos\beta\sin\gamma \\ -\sin\beta\cos\gamma & -\sin\delta\sin\gamma + \cos\delta\cos\beta\cos\gamma & \cos\delta\sin\gamma + \sin\delta\cos\beta\cos\gamma \\ \cos\beta & \cos\delta\sin\beta & \sin\delta\sin\beta \end{bmatrix} \begin{pmatrix} M_G \\ M_O \\ M_S \end{pmatrix}$$

When CMGs are used to steer spacecraft, at least three CMGs are needed to provide 3 axes control. If we consider an array of SGCMG mounted on the surfaces of a pyramid with their gimbal axes directions ($\underline{m}_i$) perpendicular to the corresponding surface and the momentum direction (h_i) always aligned with the surface of the pyramid as the gimbal σ_i rotates. The output torque from each CMG is equal to the rate of change of angular momentum which is in the ($\underline{m}_i$ x $\underline{h}_i$) direction and proportional to the gimbal rate $\dot{\delta}_i$. From the pyramid surfaces orientations we can calculate some important matrices that will be used in the equations of motion.

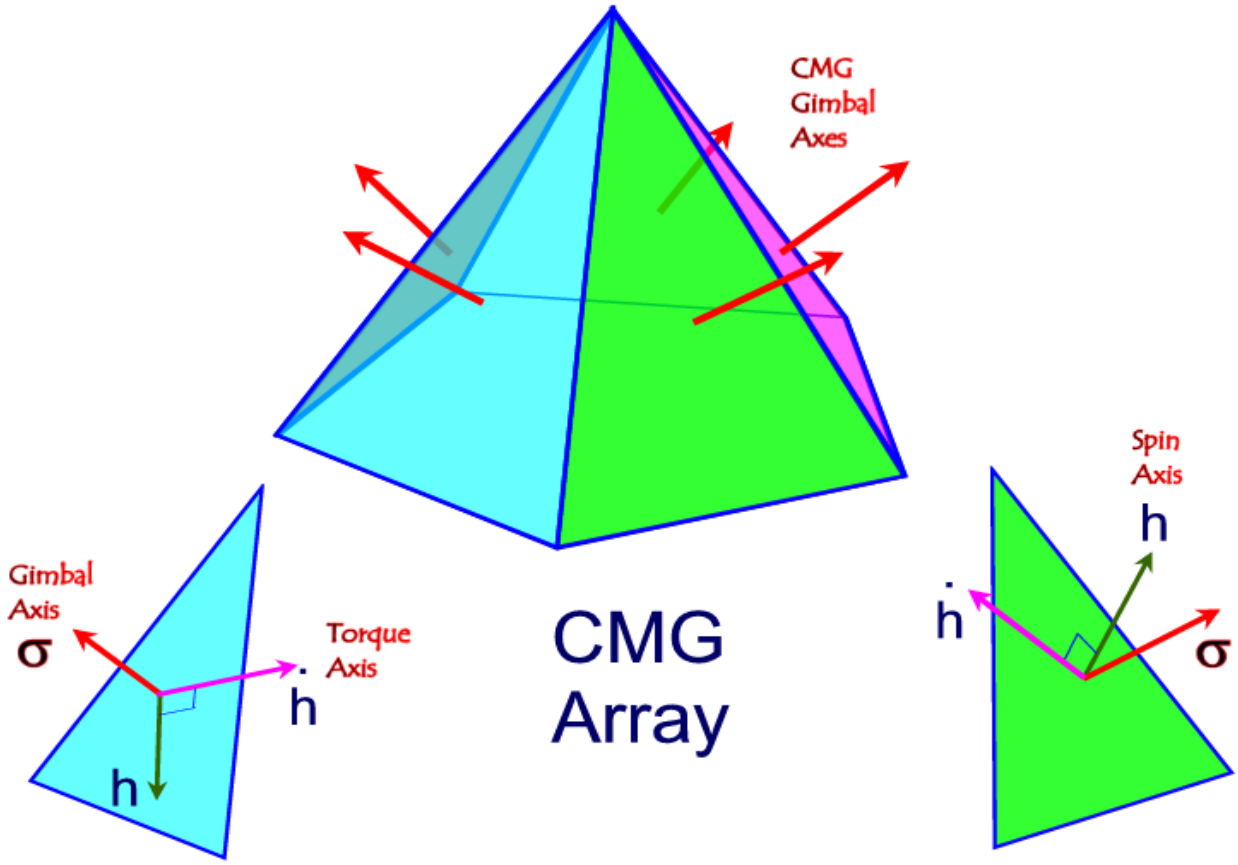


Figure 3.4 Array of five CMGs in a Pyramid Configuration

Let us consider the CMG pyramid arrangement presented in this example shown in Figure 3.5. The spacecraft has four SGCMGs which are mounted to the four faces of a four sided pyramid. All CMGs have the same angular momentum, h_{cmg}=1200 (ft-lb-sec) about their spin axis. Their CMG momentum vectors ($\underline{h}_i$) can be rotated about the gimbal vectors ($\underline{\delta}_i$), which are perpendicular to each surface, and they are constrained to lay parallel to the surface of the pyramid. The pyramid angle β is 68 (deg), and the γ_i angles of the four surfaces, according to figure (3.3), are: (90º, 180º, 270º, and 0º). The columns of the following (3x4) matrix M_g^b contains the four gimbal direction unit vectors $\underline{m}_i$. It is a gimbal to body transformation matrix.

$$M = \begin{bmatrix} \underline{m}_1 & \underline{m}_2 & \underline{m}_3 & \underline{m}_4 \end{bmatrix} \; where: \; \underline{m}_i = \begin{bmatrix} \sin\beta_i \sin\gamma_i \\ -\sin\beta_i \cos\gamma_i \\ \cos\beta_i \end{bmatrix}$$

$$M = \begin{bmatrix} \sin\beta & 0 & -\sin\beta & 0 \\ 0 & \sin\beta & 0 & -\sin\beta \\ \cos\beta & \cos\beta & \cos\beta & \cos\beta \end{bmatrix} = \begin{bmatrix} 0.927 & 0 & -0.927 & 0 \\ 0 & 0.927 & 0 & -0.927 \\ 0.375 & 0.375 & 0.375 & 0.375 \end{bmatrix} \tag{3.5}$$

The (3x4) matrix [R] represents the momentum reference directions. That is, the initial directions $\underline{r}_i$ of the momentum vectors $\underline{h}_{cmg(i)}$. More precisely, are the momentum directions when the gimbal angles ($\underline{\delta}_i$) are at zero. The initial gimbal angles δ_{0i}=0 to provide zero momentum bias.

$$R = \begin{bmatrix} \underline{r}_1 & r_2 & \underline{r}_3 & \underline{r}_4 \end{bmatrix} \; where: \; \underline{r}_i = \begin{bmatrix} \cos\gamma_i \\ \sin\gamma_i \\ 0 \end{bmatrix}; \; R = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.6}$$

We must also define a (3x4) matrix Q, containing column vectors of the cross product direction unit vectors ($\underline{q}_i$).

$$Q = \begin{bmatrix} \underline{q}_1 & \underline{q}_2 & \underline{q}_3 & \underline{q}_4 \end{bmatrix}; \; \underline{q}_i = (\underline{m}_i \times \underline{r}_i);$$

$$Q = \begin{bmatrix} -0.375 & 0 & 0.375 & 0 \\ 0 & -0.375 & 0 & 0.375 \\ 0.927 & 0.927 & 0.927 & 0.927 \end{bmatrix} \tag{3.7}$$

Notice, that the pyramid structure is only used for visualization. The CMGs do not have to be physically mounted on the four surfaces of an actual pyramid, as in Figure 3.5, but they can be translated anywhere on the spacecraft as long as their gimbal axes ($\underline{m}_i$) and their reference momentum vectors ($\underline{r}_i$) are parallel to the directions shown in the pyramid. See, for example, the CMG cluster in Figure 3.6. The CMGs are typically mounted on a structure that it is mechanically isolated from the spacecraft by means of vibration isolation struts, as shown in Figure 3.6, that attenuate vibrations from the CMGs.

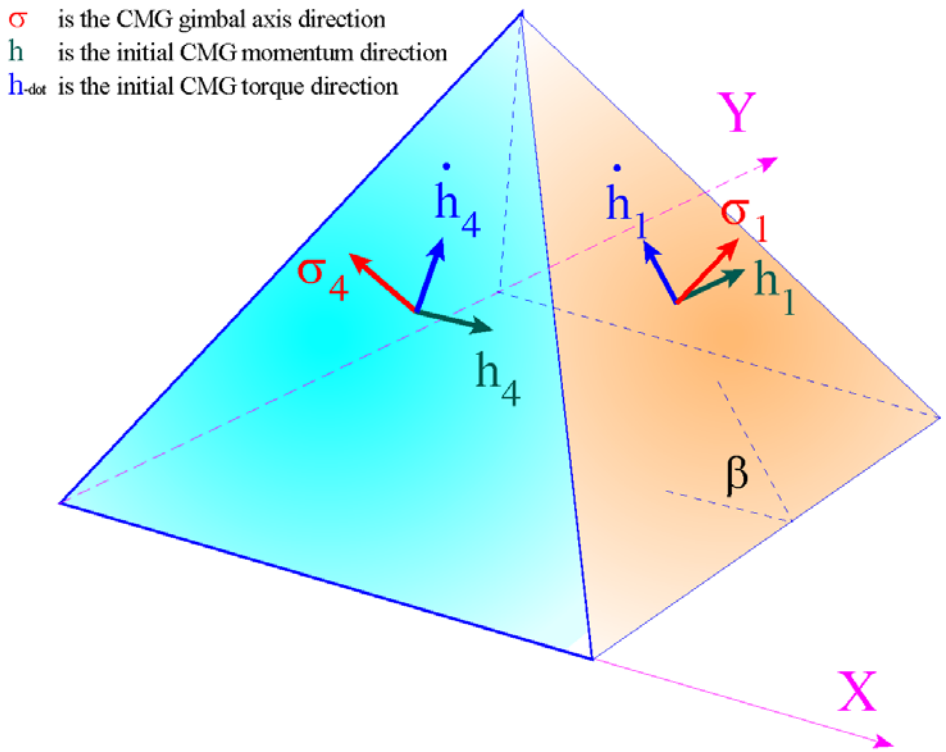


Figure 3.5 Array of four CMGs in a Pyramid Configuration

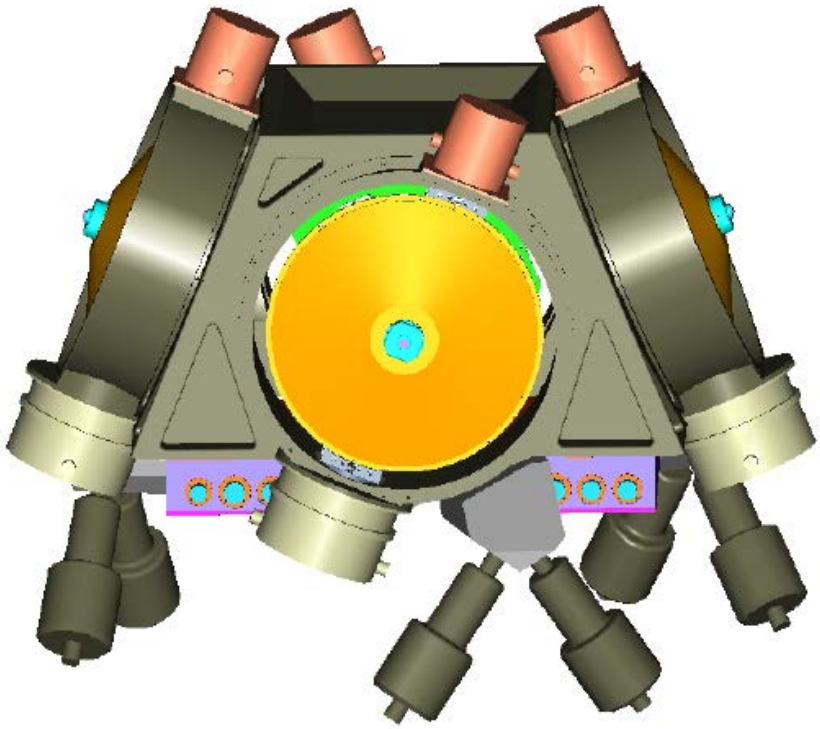


Figure 3.6 A Cluster of four Single Gimbal CMGs mounted on a pyramid structure which is isolated from the spacecraft by means of disturbance isolation struts

3.2 Dynamic Equations of a Spacecraft with SGCMG

The combined spacecraft plus CMG rate of change of momentum is related to the external torques as follows:

$$\dot{\underline{H}}_{sys} + \underline{\omega} \times \underline{H}_{sys} = \underline{T}_{ext} \qquad (3.8)$$

Where:
$\underline{H}_{sys}$ is the total system momentum,
$\underline{T}_{ext}$ is the external torque vector and
$\underline{\omega}$ is the spacecraft angular rate.

By introducing the internal CMG torque we separate the spacecraft and CMG rate of change of momentum equations and solve for the spacecraft rate $\underline{\omega}$ as a function of internal plus external torques

$$J_{sc}\dot{\underline{\omega}} = -\underline{\omega} \times J_{sc}\underline{\omega} + \underline{T}_{cmg} + \underline{T}_{ext}$$

$$\dot{\underline{H}}_{cmg} + \underline{\omega} \times \underline{H}_{cmg} = -\underline{T}_{cmg} \qquad (3.11)$$

The total system momentum is constant and it consists of spacecraft plus CMG momentum.

$$\underline{H}_{sys} = J_{sc}\underline{\omega} + \underline{H}_{cmg}$$

The internal CMG torque ($\underline{T}_{cmg}$) applied to the spacecraft is equal and opposite to the torque applied to the CMG array. It consists of the control torque ($\underline{T}_{con}$) intended to control the spacecraft which is also the rate of change in the CMG momentum, and the gyroscopic torque $\omega \times H_{cmg}$. The control torque ($\underline{T}_{con}$) is a non-linear function of the gimbal angles and gimbal rates.

$$T_{con} = -\left[A(\delta)\right]\dot{\underline{\delta}} = -\dot{H}_{cmg} \qquad (3.12)$$

The CMG array angular momentum vector in body axes ($\underline{H}_{cmg}$) is also related to the individual CMG momentum defined in CMG axes, (Gimbal, Output, and Spin axes), as follows

$$\underline{H}_{cmg} = \sum_{i=1}^{N_{cmg}} P_i \begin{pmatrix} h_G \\ h_O \\ h_S \end{pmatrix}_i \qquad (2.13)$$

Where: matrix P_i transforms the CMG (i) momentum from (Gimbal, Output, Spin) axes to spacecraft axes. Similarly, the CMG control torques are transformed from CMG axes to spacecraft axes, and they are combined to form the total CMG torque.

$$\underline{T}_{cmg} = \sum_{i=1}^{N_{cmg}} P_i \begin{pmatrix} M_G \\ M_O \\ M_S \end{pmatrix}_i \qquad (3.14)$$

where:

$$\begin{bmatrix} M_G \\ M_O \\ M_S \end{bmatrix} = \begin{bmatrix} T_{gi} \\ J_o\left(\ddot{\phi}\cos\delta - \dot{\phi}\dot{\delta}_i \sin\delta - \dot{\theta}\dot{\delta}_i \cos\delta - \ddot{\theta}\sin\delta\right) + h_0\dot{\delta}_i + \dot{\delta}_i\left(J_s - J_g\right)\left(\dot{\theta}\cos\delta + \dot{\phi}\sin\delta\right) \\ J_s\left(\dot{\Omega} + \ddot{\theta}\cos\delta + \ddot{\phi}\sin\delta + \dot{\phi}\dot{\delta}_i \cos\delta - \dot{\theta}\dot{\delta}_i \sin\delta\right) + \dot{\delta}_i\left(J_g - J_o\right)\left(\dot{\phi}\cos\delta - \dot{\theta}\sin\delta\right) \end{bmatrix}$$

$$P_i = \begin{bmatrix} \sin\beta\sin\gamma & -\sin\delta\cos\gamma - \cos\delta\cos\beta\sin\gamma & \cos\delta\cos\gamma - \sin\delta\cos\beta\sin\gamma \\ -\sin\beta\cos\gamma & -\sin\delta\sin\gamma + \cos\delta\cos\beta\cos\gamma & \cos\delta\sin\gamma + \sin\delta\cos\beta\cos\gamma \\ \cos\beta & \cos\delta\sin\beta & \sin\delta\sin\beta \end{bmatrix}$$

By combining equations (3.11) and (3.12) we can rewrite equation (3.11) in terms of only the control torque instead of the total CMG torque ($\underline{T}_{cmg}$) as shown in (3.15). T_{con} is the steering torque designed to shape the spacecraft rate as commanded by the Attitude Control System. Equations (3.15) can be used instead of (3.11) in simple 6-dof simulations that do not require gimbal torque dynamics and flexibility.

$$J_{sc}\dot{\underline{\omega}} + \left(\underline{\omega} \times \underline{H}_{sys}\right) = T_{con} + \underline{T}_{ext}$$

$$\dot{\underline{H}}_{cmg} = -\underline{T}_{con} \qquad (3.15)$$

If we consider only the CMG momentum about its spin axes and ignore the momentum about the gimbal and output axes (which are small), the combined CMG angular momentum vector is a function of the individual CMG momentums $h_{cmg(i)}$ and also a function of their spin axis orientations, as shown in equation (3.16). For a more accurate implementation that includes also the effects due to the gimbal torques we may calculate the CMG momentum ($\underline{H}_{cmg}$) by integrating equation (3.11) using T_{cmg} from (3.14).

$$\underline{H}_{cmg} = \sum_{i=1}^{N_{cmg}} \left(\cos\delta_i\, \underline{r}_i + \sin\delta_i\, \underline{q}_i\right) h_{cmg(i)} \qquad (3.16)$$

The matrix A that relates the gimbal rates to rate of change in CMG momentum is a (3 x N_{cmg}) matrix consisting of (N_{cmg}) column vectors a_i. Its elements vary with the gimbal angles (δ_i).

$$A(\delta) = \begin{pmatrix} \underline{a}_1 & \underline{a}_2 & \underline{a}_3 & \underline{a}_4 \end{pmatrix} \quad where:$$

$$a_i = \left(\cos\delta_i\, \underline{q}_i - \sin\delta_i\, \underline{r}_i\right) h_{cmg(i)} \qquad (3.17)$$

δ_i is the gimbal angle for CMG (i)
$\underline{r}_i$ is a unit vector of the initial momentum direction for CMG (i)
$\underline{m}_i$ is a unit vector of the gimbal direction for CMG (i)
$\underline{q}_i$ is the orthogonal direction (m_i x r_i) for CMG (i)
T_{con} is the control torque applied to the gimbal by the motor
T_{gi} is the gyroscopic torque $-\omega \times H_{cmg}$ resolved in gimbal (i) direction
M_g^b is the (3 x 4) transformation matrix from gimbal axis to body axis
$h_{cmg(i)}$ is the momentum of CMG (i) about its spin axis, which is constant.

The CMG gimbal rates are controlled by a servo system that generates gimbal torques. The servo torque at the gimbal of each CMG is attempting to counteract the gyroscopic disturbance torque created by the spacecraft rate. $\dot{\theta}$ *and* $\dot{\phi}$ are the spacecraft rates resolved about the CMG reference and quad axes, as defined in equation (3.3). The CMG gimbal inertial acceleration is obtained by integrating the gimbal moment equation (3.18) ignoring friction. T_{gi} is the motor torque applied at each gimbal (i). Even though the CMG moment of inertia about the gimbal J_g is relatively small, the $\omega \times h_{cmg}$ gyroscopic moment caused by the CMG momentum coupling with spacecraft rate is a big torque that requires a powerful gimbal servo-motor in order to be able to control the gimbal rate.

$$J_g \ddot{\delta}_i + h_{cmg(i)}\left(\dot{\theta}\sin\delta - \dot{\phi}\cos\delta\right) = T_{gi} \qquad (3.18)$$

The attitude quaternion is updated by integrating the quaternion rate which is a function of the body rate and the current quaternion, as shown in equation (2.19)

$$\dot{\underline{Q}} = 0.5\begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix}\underline{Q} \qquad (3.19)$$

The pseudo-inverse of matrix A is used to calculate the SGCMG steering law

$$\dot{\delta}_{com} = -A(\delta)^{+}\left(J_{sc}\dot{\omega}_{com} + \left(\omega \times H_{sys}\right)_{estim}\right) \qquad (3.20)$$

After substituting in equation (2.12) it makes the vehicle rate to be equal to the commanded rate.

$$\dot{\omega} = \dot{\omega}_{com}$$

Linearized Equations of Spacecraft with SGCMGs

The rate of change in CMG momentum is shown in equation (3.21), where: (ω_{X0}, ω_{Y0}, ω_{Z0}) is the nominal (steady) body rate and (ω_x, ω_y, ω_z) is the variation in vehicle rate. Similarly, (H_{X0}, H_{Y0}, H_{Z0}) is the nominal (steady) CMG array momentum and (h_x, h_y, h_z) is the variation in CMG momentum. $\underline{T}_{cmg}$ is the torque applied to the vehicle generated by the CMG array.

$$\begin{pmatrix} \dot{h}_x \\ \dot{h}_y \\ \dot{h}_z \end{pmatrix} = -\begin{pmatrix} \omega_{X0} \\ \omega_{Y0} \\ \omega_{Z0} \end{pmatrix} \times \begin{pmatrix} h_x \\ h_y \\ h_z \end{pmatrix} - \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \times \begin{pmatrix} H_{X0} \\ H_{Y0} \\ H_{Z0} \end{pmatrix} - \underline{T}_{cmg} \tag{3.21}$$

The CMG torque as a function of the spacecraft rates, and the gimbal angles and rates is:

$$\underline{T}_{cmg} = -\sum_{i=1}^{N_{cmg}} P_i \begin{bmatrix} T_{gi} \\ h_{cmg(i)}\dot{\delta}_i + \left(J_s - J_g\right)\left(\dot{\theta}_0 \cos\delta_0 + \dot{\phi}_0 \sin\delta_0\right)\dot{\delta}_i \\ \left(J_g - J_o\right)\left(\dot{\phi}_0 \cos\delta_0 - \dot{\theta}_0 \sin\delta_0\right)\dot{\delta}_i \end{bmatrix}_i \tag{3.22}$$

The CMG gimbal accelerations in the linearized model are obtained by linearizing equation (2.18)

$$J_g \ddot{\delta}_i = T_{gi} - h_{cmg(i)}\left(\dot{\theta}_0 \cos\delta_0\, \delta + \dot{\phi}_0 \sin\delta_0\, \delta + \sin\delta_0 \dot{\theta} - \cos\delta_0 \dot{\phi}\right) \tag{3.23}$$

where:

$\left(\dot{\theta}_0, \dot{\phi}_0, \dot{\delta}_0\right)$ are the nominal values of vehicle and gimbal rates

$\left(\dot{\theta}, \dot{\phi}, \dot{\delta}\right)$ are the variations in vehicle and gimbal rates

SGCMG References:

1. A Practical Approach to Modeling Single Gimbal CMGs in Agile Spacecraft, Chris J. Heiberg, AIAA GN&C Conference, August 2000
2. Precision Spacecraft Pointing Using Single Gimbal CMGs, Chris Heiberg, Dave Bailey, Bong Wie, Journal of Guidance, Control, and Dynamics, Vol. 23, No. 1, January–February 2000
3. Singularity Robust Steering Logic for Redundant Single-Gimbal Control Moment Gyros, Bong Wie, Chris Heiberg, Dave Bailey, AIAA GN&C Conference, August 2000, AIAA-2000-4453
4. Feedback Control Law for Variable Speed Control Moment Gyros, Hanspeter Schaub, Srinivas R. Vadali, John L. Junkins, Journal of the Astronautical Sciences, Vol. 46, No. 3, July–Sept., 1998
5. Rapid Multi-Targeting Acquisition and Pointing Control of Agile Spacecraft, Bong Wie, Dave Bailey, Chris Heiberg, AIAA GN&C Conference, August 2000, AIAA-2000-4546
6. Precision Pointing Control of Agile Spacecraft using Single Gimbal CMG, Chris Heiberg, Dave Bailey, Bong Wie, AIAA-97-3757
7. Bong Wie, et. al., "Singularity Robust Steering Logic for Single-Gimbal Control Moment Gyro," J. of GCD, vol. 23, No. 5, Sep-Oct, 2001.
8. Bong Wie, *Space Vehicle Dynamics and Control*, AIAA, 2nd edition, 2010.

3.3 Maximum Energy Control

The momentum exchange devices, such as Control Moment Gyros or Reaction Wheels have limits in torque and momentum capability. Linear control laws are too slow for agile spacecraft maneuvering because they do not utilize the max torque and momentum capability of the device. For fast maneuvering between targets the spacecraft control law should be able to utilize the max torque and momentum capability of the momentum exchange device.

Let us consider the rectangular torque profile for a bang-bang, time optimal maneuver shown in Figure 2.3.1(a). A desired slew maneuver usually requires two torque pulses, one to initiate the maneuver and the other to stop the maneuver once the desired angle has been achieved. Using the figure, the maneuver time t_s, the maximum torque magnitude T_r, and the maneuver angle θ, are related as follows:

$$T_r = I\ddot{\theta} = \frac{4I}{t_s^2}\theta \qquad (3.3.1)$$

Where:

$$\ddot{\theta} = \frac{2}{t_s}\dot{\theta}_m \qquad (3.3.2)$$

is the acceleration expressed in terms of the maximum rate, $\dot{\theta}_m$, and in terms of the spacecraft moment of inertia *I*, about the maneuvering axis. In a typical bang-bang maneuver we have two types of situations: (a) when you apply max torque during half of the maneuvering time and then reverse and apply negative torque during the second half of the maneuver, without the spacecraft rate reaching the max momentum capability of the device, and (b) when you apply max torque for a period t1 until the momentum device reaches near saturation, followed by a zero torque coasting period (t1 to t2), and at t2 the torque is reversed to max negative value and applied for the same amount of time (t2 to t3) until the spacecraft reaches target position.

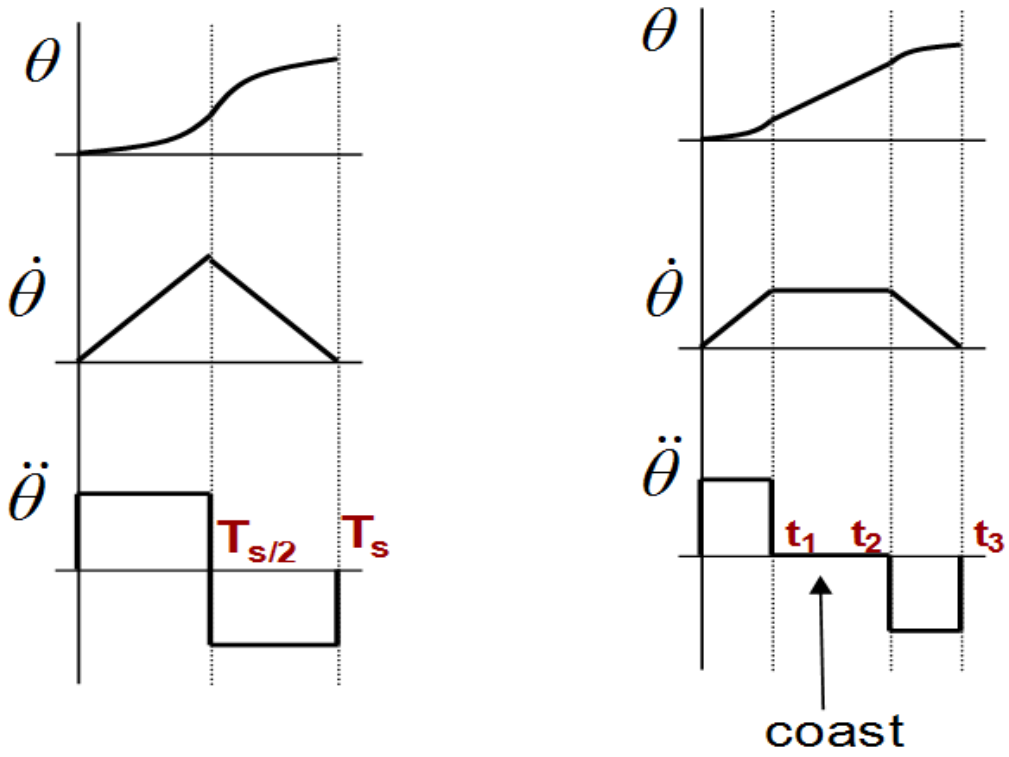


Figure 2.3.1 Bang-Bang Maneuvers

The maximum energy control is a phase-plane control law that attempts to perform a similar type of bang-bang fast maneuvering in three directions. It operates in two distinct modes to achieve good performance; the phase-plane mode, and the PID mode, and it uses a switching logic to switch between the two modes. The phase-plane mode provides the fast maneuvering feature, while at steady-state the PID provides good tracking capability.

During maneuvering the logic calculates the quaternion error between current and the desired target positions and performs the maneuver in the quaternion error eigenaxis. The phase-plane maneuvering logic uses a three-degree-of-freedom switching line to converge the attitude versus rate error trajectory to zero (target position). The switching line is calculated based on the maximum acceleration capability of the CMG. In the beginning of the maneuver the spacecraft uses maximum CMG torque in the error direction and it reverses the torque when the trajectory reaches the switching line. Then it follows the switching trajectory to zero. There are limits, of course, that prevent the spacecraft rate from exceeding max rate and acceleration limits. When the spacecraft approaches near the target position the logic switches to PID mode which provides good tracking by further minimizing the attitude error. Maintaining a constant eigenaxis creates a very stable maneuver. The attitude errors are maintained proportional in all directions and they are all brought to zero in unison.

Maneuvering Phase-Plane Logic

For a single axis rotation the control torque switching line in the rate versus attitude error phase-plane is as follows:

$$\dot{x} = \sqrt{-2a_{lim}x} \quad \text{when} \quad x < 0$$

$$\dot{x} = -\sqrt{2a_{lim}x} \quad \text{when} \quad x > 0$$

where: $\mathbf{a}_{lim}$ is the acceleration limit and it is defined by the ratio of maximum torque over inertia in that direction:

$$a_{lim} = \frac{T_{max}}{I}$$

For a three axes rotation, the phase plane switching line becomes

$$\dot{x}_i = \sqrt{\frac{2x_i^2}{\sqrt{\frac{x_1^2}{A_1^2} + \frac{x_2^2}{A_2^2} + \frac{x_3^2}{A_3^2}}}} \qquad (3.3.3)$$

Where: A_1, A_2, and A_3 are the acceleration limits in spacecraft body directions.

Attitude Control System Description

The maximum energy control system block diagram is shown in Figure 3.3.2. It consists of an inner rate control loop (D) and an outer (PI) attitude control loop. The CMG array steering logic is also included in the rate control loop. The inner rate control loop must be designed first. It receives a body rate command from the attitude controller and regulates the vehicle rate by issuing acceleration commands to the RW steering logic. If the bandwidth of the rate loop is sufficiently high to keep up with the body rate command then we can assume that ($\omega \approx \omega_c$) and continue with the outer loop.

Inner Rate Loop

The rate control loop regulates the spacecraft rate. It receives (roll, pitch, and yaw) rate error commands which become acceleration commands that drive the CMG steering law. The steering law generates the CMG gimbal rate commands ($\dot{\delta}_{com}$). An acceleration limiter (AL) is included in the rate loop that prevents the CMG torque commands from exceeding their maximum torque capability. The steering law is given in equation (3.3.4). It uses a pseudo-inverse of matrix A which is defined in equation 3.15 and it is a function of the gimbal angles δ_i

$$\underline{\dot{\omega}}_{com} = k_r \, \underline{\omega}_{err}$$

$$if \left|\underline{\dot{\omega}}_{com}\right| > AL, \quad \underline{\dot{\omega}}_{com} = \underline{\dot{\omega}}_{com} \frac{AL}{\left|\underline{\dot{\omega}}_{com}\right|} \tag{3.3.4}$$

$$\underline{\dot{\delta}}_{com} = -\left[A^T \left(AA^T\right)^{-1} + \lambda E\right] J \, \underline{\dot{\omega}}_{com}$$

Depending on the orientation of the gimbal angles δ_i the pseudo-inverse matrix may become singular. To avoid the singularity, the pseudo-inverse matrix is modified using a singularity avoidance logic (λE), where:

$$E = \begin{bmatrix} 1 & 0.01\sin(wt) & 0.01\cos(wt + \frac{\pi}{2}) \\ 0.01\sin(wt) & 1 & 0.01\sin(wt - \frac{\pi}{2}) \\ 0.01\cos(wt + \frac{\pi}{2}) & 0.01\sin(wt - \frac{\pi}{2}) & 1 \end{bmatrix} \quad and \quad \lambda = \frac{k}{\det(AA^T)}$$

Where: w and *k* are properly selected. The singularity avoidance logic checks the condition of the A matrix and introduces a small perturbation to prevent a gimbal lock. When the matrix A is moving towards a singularity the perturbation rotates around it. The closer to a singularity the bigger the perturbation gets.

Attitude Control Loop

The input to the attitude control loop is the attitude error, or more precisely, quaternion error. The attitude error signal goes to the PI controller that operates in two modes based on the switch k_y which is either set to zero or one. During maneuvering (when k_y=1) the controller becomes a simple proportional gain K_p. Otherwise, during PID mode, (that is when k_y=0) the block becomes a PI transfer function $K_P \frac{s+b}{s}$ that provides better tracking and disturbance attenuation at low frequencies. The switch setting is controlled by a signal that is a combination of rate plus attitude error. In the beginning of the maneuver when this error signal is large k_y is set to 1. When the error signal drops below a certain value k_y is set to zero and the integrator is turned on.

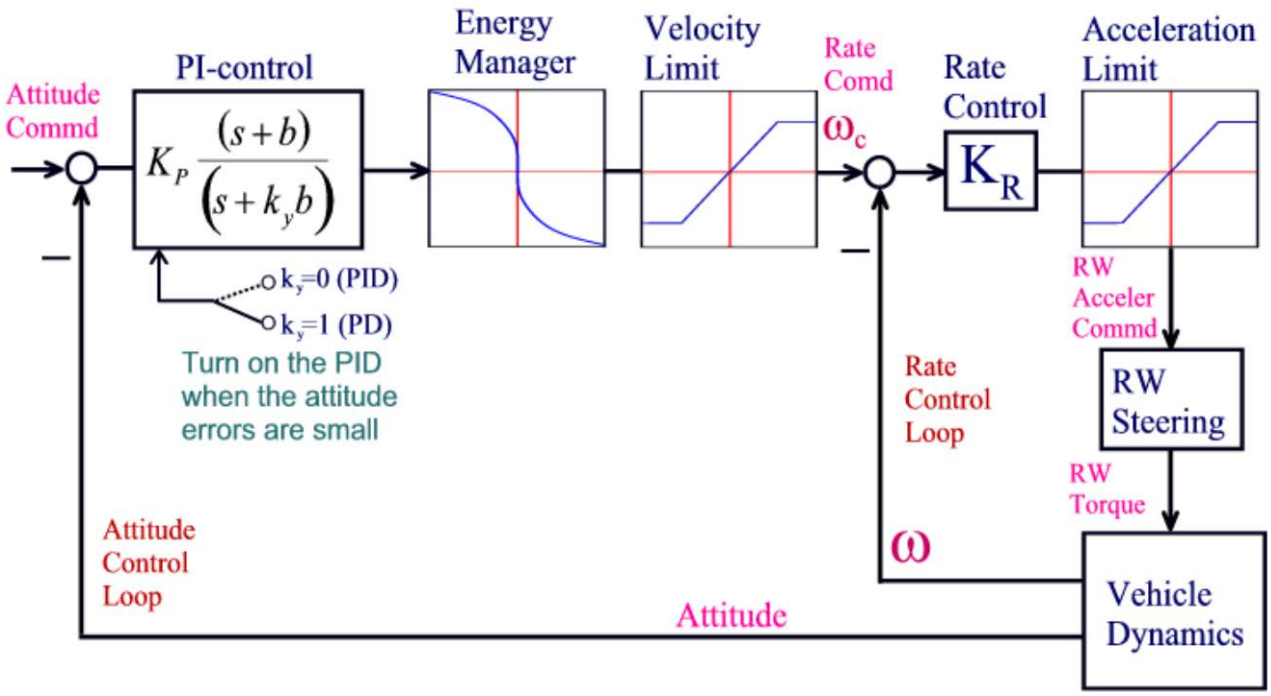


Figure 3.3.2 Block Diagram of Maximum Energy Control System

The energy manager block shapes the velocity command as a function of the attitude error in order to bring the phase-plane trajectory directly to zero without chattering, according to the parabolic switching line in equation (3.3.3). This maintains a proportional attitude error reduction in all directions throughout the maneuver. The velocity limiter in series with the energy limiter bounds the vehicle rate during the acceleration phase of the maneuver. It also bounds the CMG momentum during maneuvering preventing it from reaching saturation levels.

3.4 Simulation Models

In this section we are presenting simulations and analysis of the agile spacecraft controlled by an array of four SGCMG. We start with a simple rigid-body simulation model that uses equations (2.13) and is assuming that the gimbal rates are equal to the gimbal rate commands coming from the steering logic. The CMG momentum is calculated from equation (3.14) as a function of the gimbal angles. The gimbal rate commands are calculated by the steering logic which was described in equation 3.3.4. The control torque on the spacecraft is $\underline{T}_{con} = -[A(\delta)]\dot{\underline{\delta}}$. This model is used for initial evaluation of the control law before advancing into more complex models. The second simulation model is still rigid-body uses equations (3.11) and (3.12). The CMG torque in addition to the control torque T_{con} it contains also an estimate of the gyroscopic torques. The CMG momentum is calculated by integrating the second part of equation (3.11). The CMG torques are calculated from equations (3.2), (3.4) and (3.10). The gimbal rates are calculated by integrating equation (3.16). The gimbal torque Tgi is provided by a gimbal torque motor servo system. The quaternion output is obtained by integrating equation (3.17). The third simulation model is similar to the second one but it was adjusted to include structural bending.

3.4.1 Max-Energy/ 4 CMG Simple Rigid Body Simulation

Figure 3.4.1.1 shows the Max-Energy Attitude Control System simple simulation model that uses four Single-Gimbal CMGs. It is implemented in a Simulink file "*MaxEn-NonLin_4SGCMG.mdl*" which is located in folder "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(a) Simple RigBody 4SGCMG ACS*". It consists of three major blocks: the Attitude Control System, the inner rate loop and Steering logic block, and the spacecraft/ CMG dynamics. The ACS receives a quaternion command which is compared with the spacecraft quaternion attitude to generate the error signal which drives the Steering logic and rotates the spacecraft.

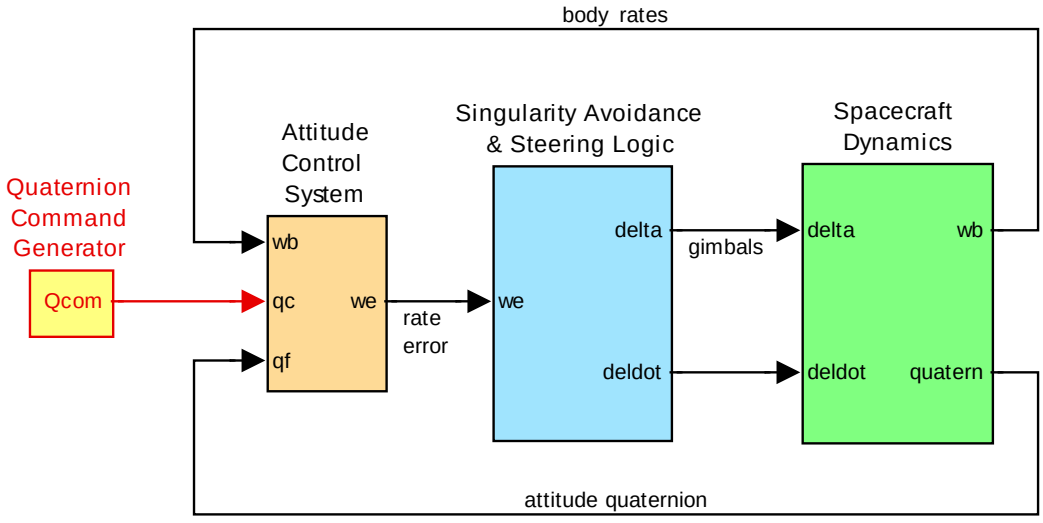
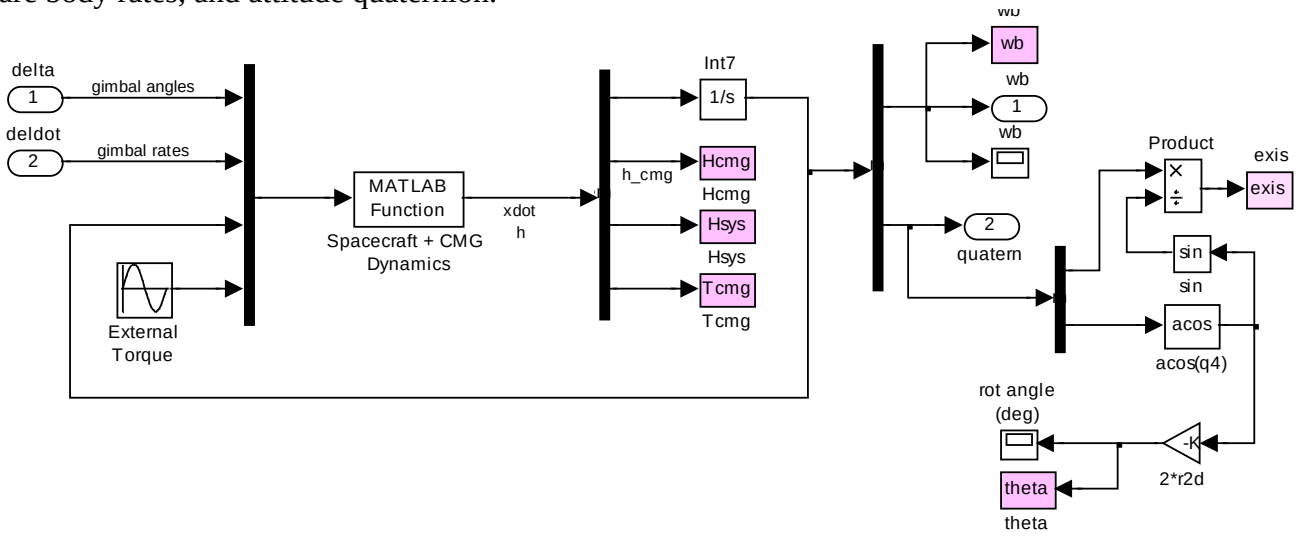


Figure 3.4.1.1 Max-Energy ACS Simulation Model "MaxEn_NonLin_4SGCMG.mdl"

The spacecraft and CMG dynamics (green block) is implemented in Matlab function "*CMG_Dynamics.m*". The inputs are CMG gimbal angles and gimbal rates in (radian). The outputs are body rates, and attitude quaternion.




```
function dot= CMG_Dynamics(delta,deldot,x,Td)          % s/c Dynamics with CMG
global J Jinv m ref quad hcmg d2r r2d wcmg zeta

% State Variables (x)
% x(1-3)   = Body rates   (w)   (rad/sec)
% x(4-7)   = Quaternion
% Inputs:
% Td(3)    = Disturbance Torque (ft-lb)
% delta(4) = Gimbal Deflections (rad)
% deldot(4)= Gimbal Rates (rad/sec)

dot= zeros(16,1);
w= x(1:3);
h= [0 0 0]';

for i=1:4
    A(:,i)=     (-sin(delta(i))*ref(:,i) + cos(delta(i))*quad(:,i))*hcmg(i);
    h = h +      (cos(delta(i))*ref(:,i) + sin(delta(i))*quad(:,i))*hcmg(i);
    harray(:,i)= (cos(delta(i))*ref(:,i) + sin(delta(i))*quad(:,i))*hcmg(i);
end

hs = J*w + h;                                          % System Momentum
Tcmg = -A*deldot -cross(w,h);                          % CMG Control Torque
dot(1:3)= Jinv*(Tcmg - cross(w,J*w) +Td);              % Vehicle acceleration
dot(4:7)= 0.5*[0      w(3)  -w(2)   w(1);              % Quaternion Update
              -w(3)   0      w(1)   w(2);
               w(2)  -w(1)   0      w(3);
              -w(1)  -w(2)  -w(3)   0]*x(4:7);
dot(8:10) = h;                                         % CMG Momentum in body
dot(11:13)= hs;                                        % System Momentum
dot(14:16)= Tcmg;                                      % CMG Torques
```

The inner rate control loop and Steering Logic (middle block in Figure (3.4.1.2) is implemented in Matlab function "*Steering.m*". It receives the rate error command from ACS and calculates the gimbal rate command $\dot{\delta}_{com}$.

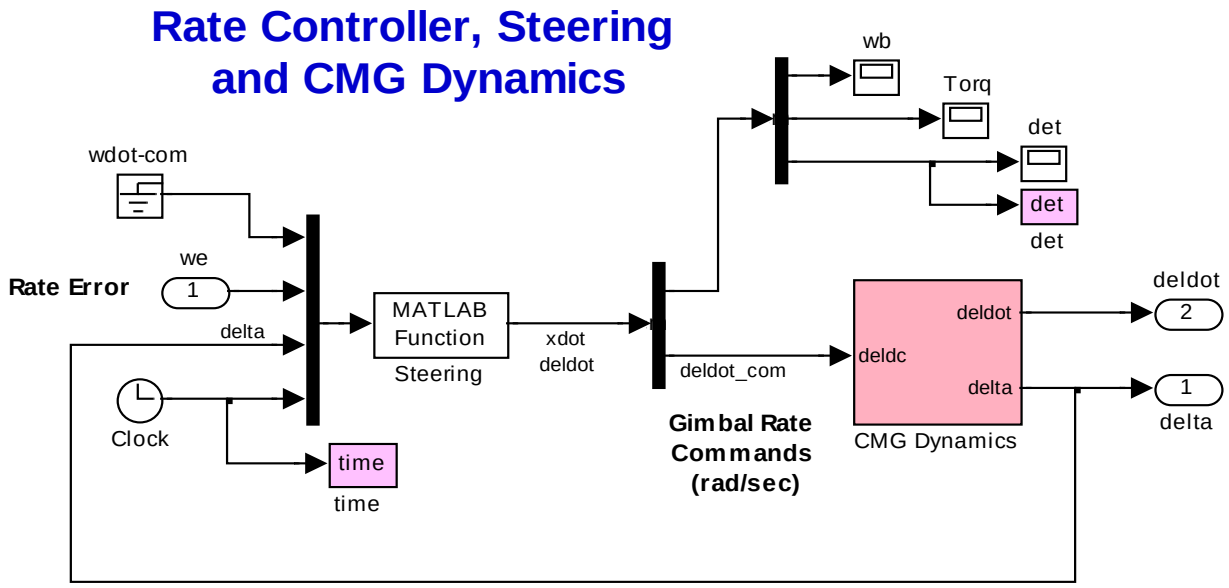
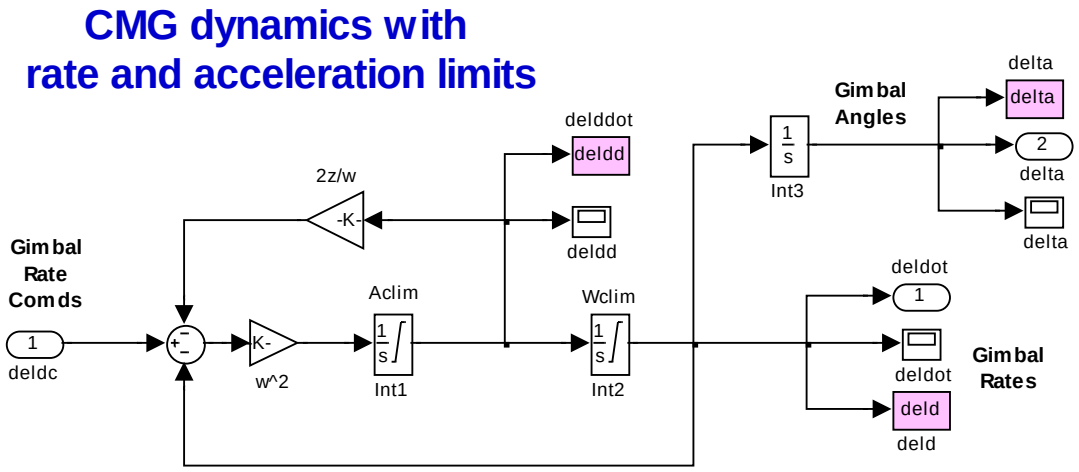


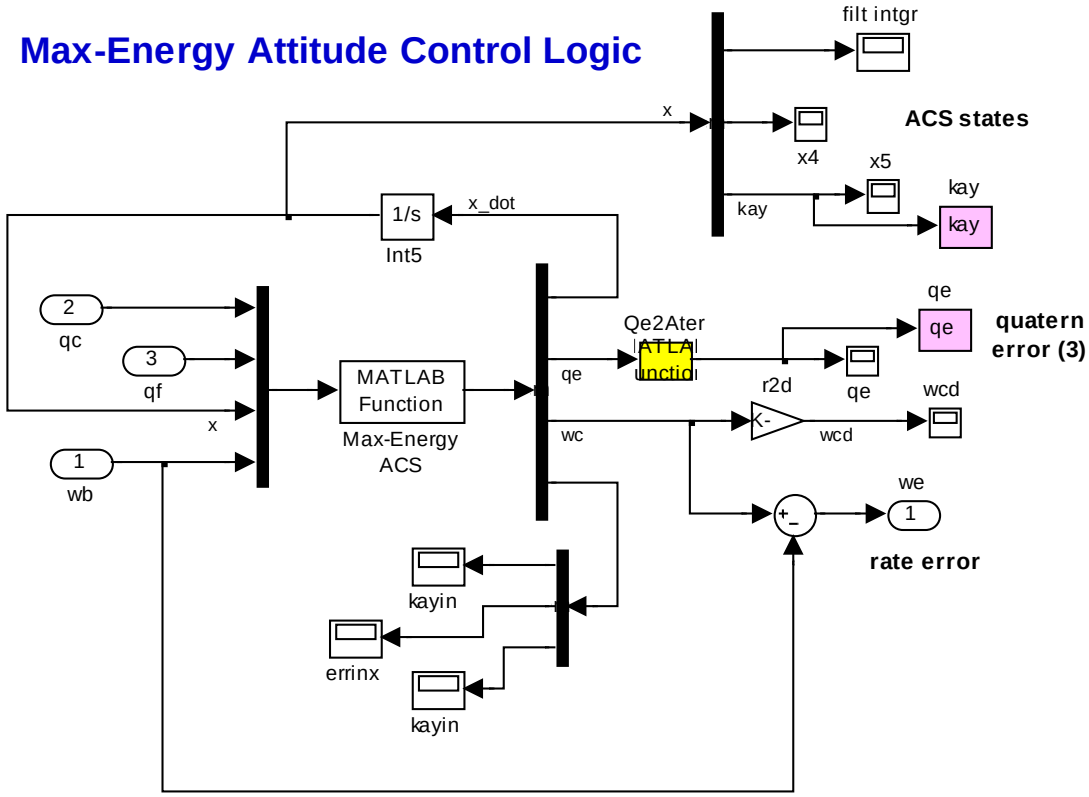
Figure 3.4.1.2 Inner Rate Loop and Steering Logic

```
function dot= Steering(wdcom,we,delta,t)
global J m ref quad hcmg d2r r2d wcmg zeta
global Acc_Lim Rat_Lim kr Es Ps Fs
%
% State Variables (x)
% x(1-3)  = Jerk Limiter
% we(3)   = Vehicle rate error
% delta(4)= Gimbal angles

dot= zeros(7,1);                                % 7 outputs

for i=1:4
   A(:,i)=(-sin(delta(i))*ref(:,i) + cos(delta(i))*quad(:,i))*hcmg(i);
end
wdot= kr*we;                                    % rate error

% Variable Acceleration Limit
wdotlim=  sqrt(Acc_Lim*sqrt(we'*we));
wdotrate= sqrt(wdot'*wdot);
if wdotrate>wdotlim; wdot=wdot*wdotlim/wdotrate; end

% Fixed Rate Limit
wdotmag= sqrt(wdot'*wdot); dot(1)= wdotmag;
if wdotmag > Rat_Lim; wdot= wdot*Rat_Lim/wdotmag; end
hdot = J*wdot; dot(2)= sqrt(hdot'*hdot);            % Desired torque

% Singularity avoidance
dot(3)=det(A*A'); lamb= Ps/det(A*A');
E= [1               Es*sin(Fs*t)      Es*cos(Fs*t+pi/2);
   Es*sin(Fs*t)      1               Es*sin(Fs*t-pi/2);
   Es*cos(Fs*t+pi/2)  Es*sin(Fs*t-pi/2)      1];

pinverse= A'*inv(A*A' + lamb*E);
dot(4:7)=    -pinverse*hdot;                                  %  Delta  dot  command
```

The CMG dynamics block captures the CMG second order dynamics and the gimbal rate and acceleration limits. The outputs from the CMG dynamics block are CMG gimbal rates and gimbal angles, $(\dot{\delta}_i, \delta_i)$.



The ACS Max-Energy ACS block is shown expanded below and its logic is implemented in function "*Max_Energy_ACS.m*". The algorithm generates a vehicle rate command (w_c) from which the body rate (w_b) is subtracted to calculate the rate error (w_{err}) which commands the inner rate control loop. The PID integrator switching logic is also implemented in this function.



Simulation Results

The initialization file "Start.m" initializes the Simulink models before running the simulations. This m-file loads the spacecraft mass properties, CMG parameters, and a transformation matrix Tc2b from CMG to body coordinates. It calls function M4.m to calculate the matrices M, R, and Q. It defines the CMG gimbal system bandwidth, damping, plus gimbal rate and acceleration limits. It defines also spacecraft ACS and steering max rate limits and max accelerations, ACS gains, the singularity avoidance parameters, and the rotation command eigenaxis. The rotation angle command is defined inside the quaternion command yellow block in the simulation model. The simulation results in Figure (3.4.1.3) show the ACS response to a 90° maneuver.

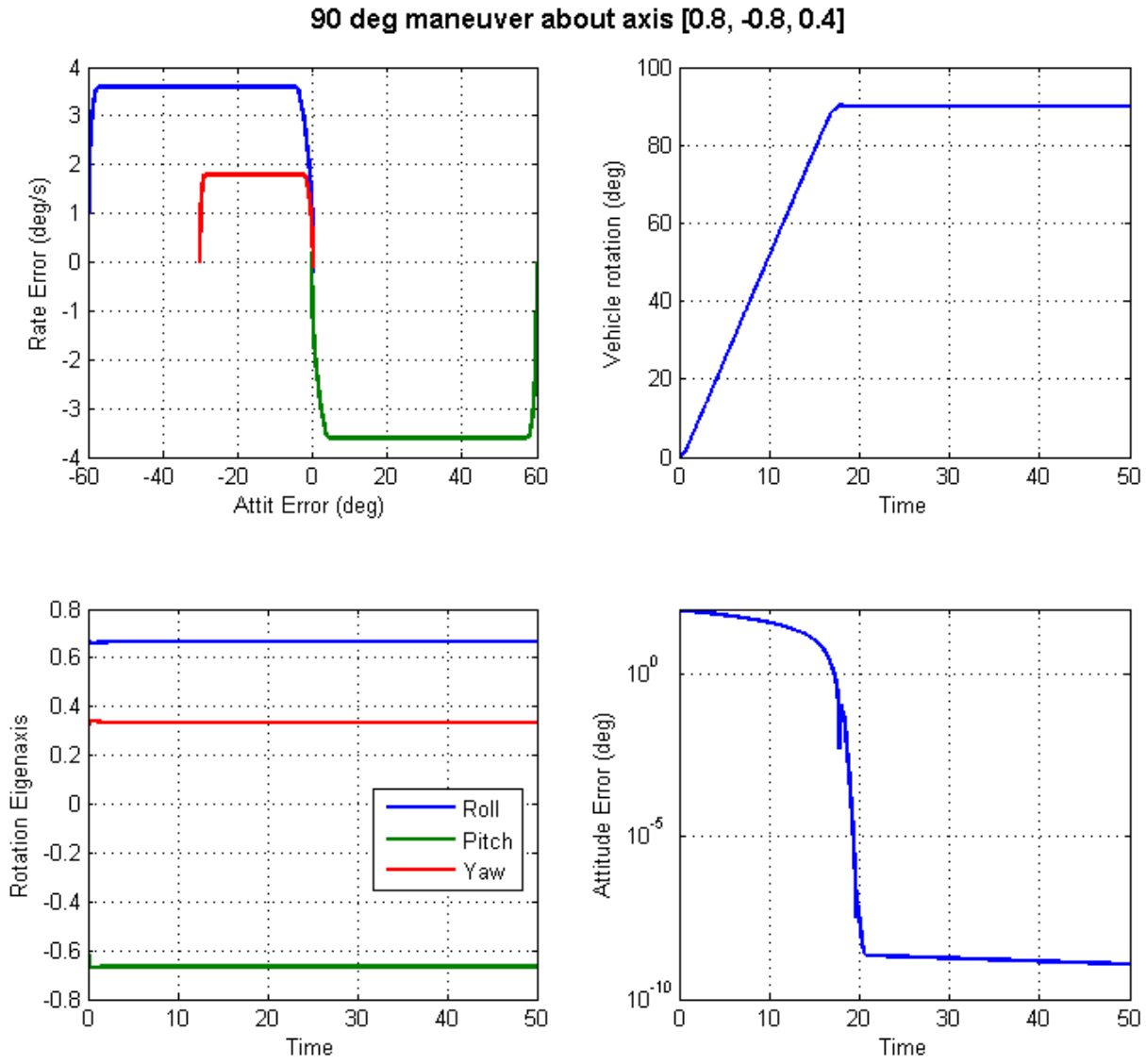


Figure 3.4.1.3(a) System performs a perfect rotation about the commanded eigenaxis achieving very small attitude errors in 20 seconds.

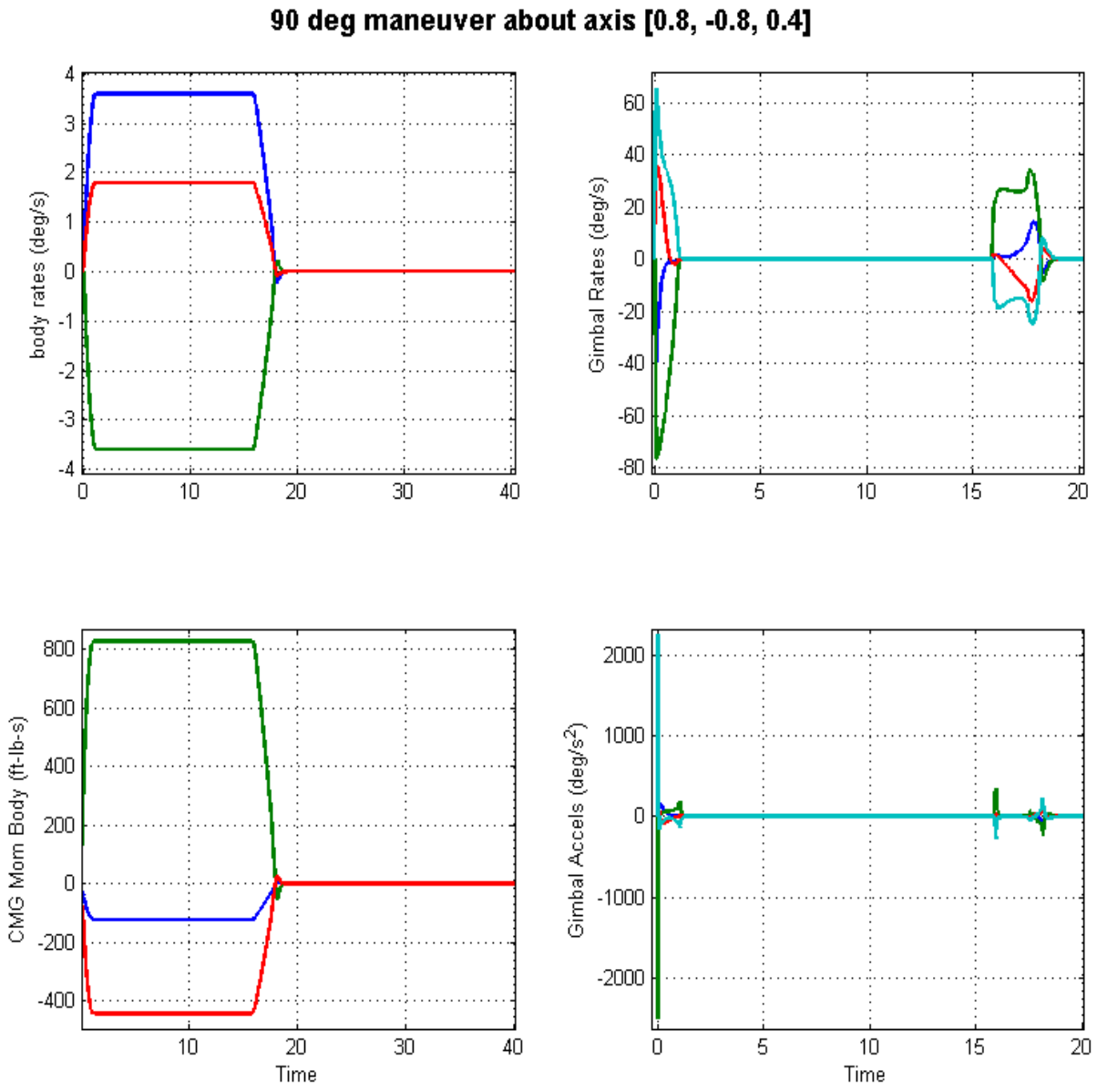


Figure 3.4.1.3(b) Max acceleration is used to reach max momentum capability where the vehicle maintains constant rate until acceleration is reversed to slow it down to its target position.

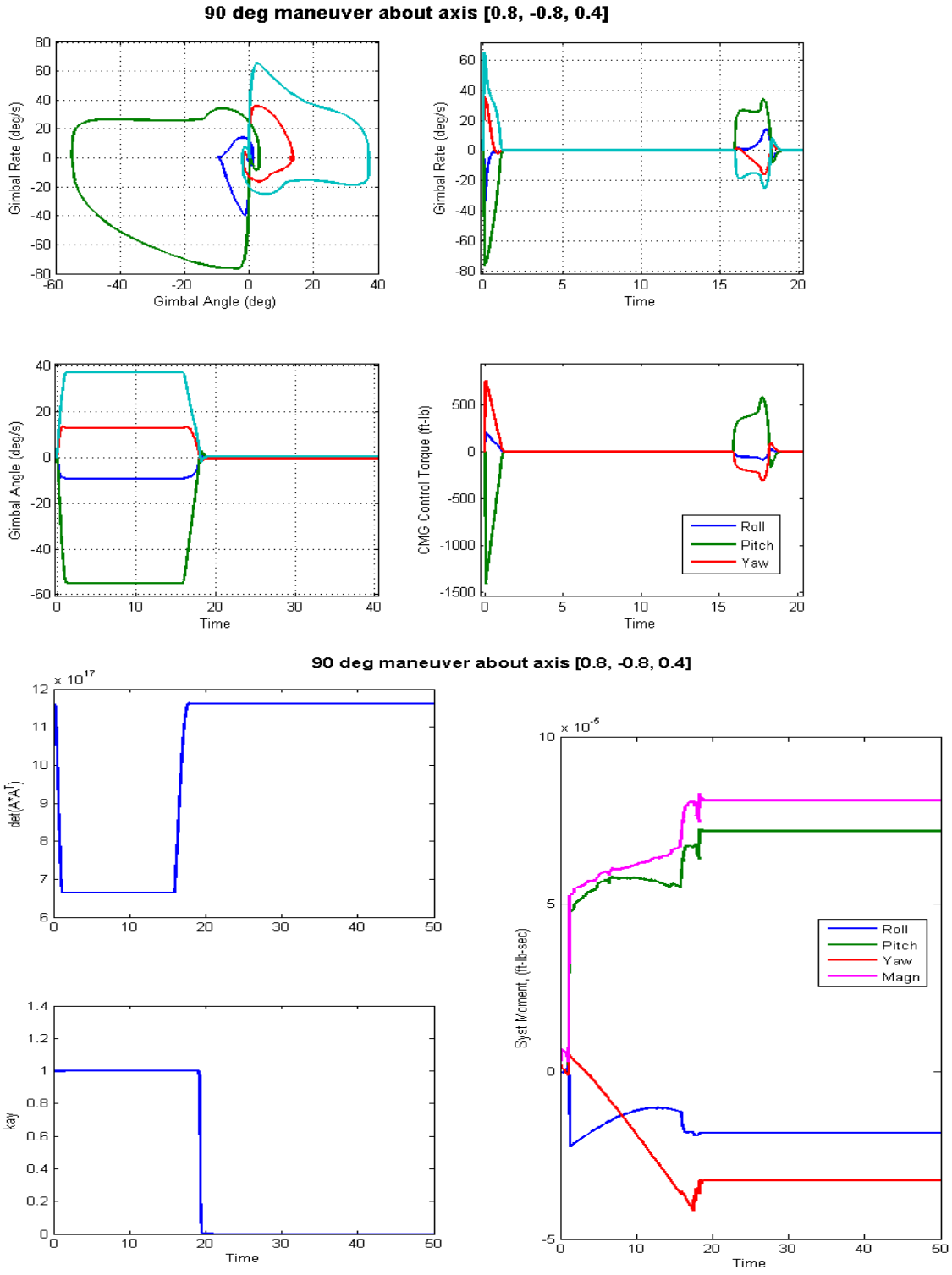


Figure 3.4.1.3(c) System momentum remains constant (zero) throughout the maneuver. At approx. 20 seconds the PID integrators are turned on to further attenuate attitude errors. The gimbal angles are at constant positions during the middle section of the maneuver maximizing the spacecraft momentum along the commanded eigenaxis. The gimbal rates and torques are zero when the momentum is constant.

Linear Frequency Domain Analysis

The file “freq.m” performs linear analysis in the frequency domain. It linearizes two Simulink models. The open-loop model “Open-Loop.mdl” in figure (2.4.1.4) is used to calculate the frequency response and the phase/ gain margins shown highlighted red in figure (2.4.1.5). In the following example we calculate the frequency response across the open yaw loop with the pitch and roll loops closed.

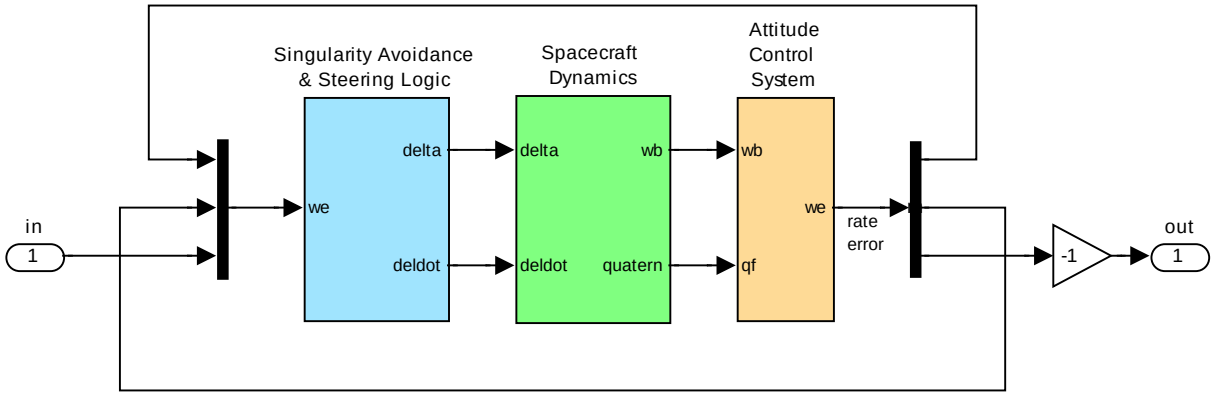


Figure 3.4.1.4 Linear Model “Open_Loop.mdl” used for Stability Analysis

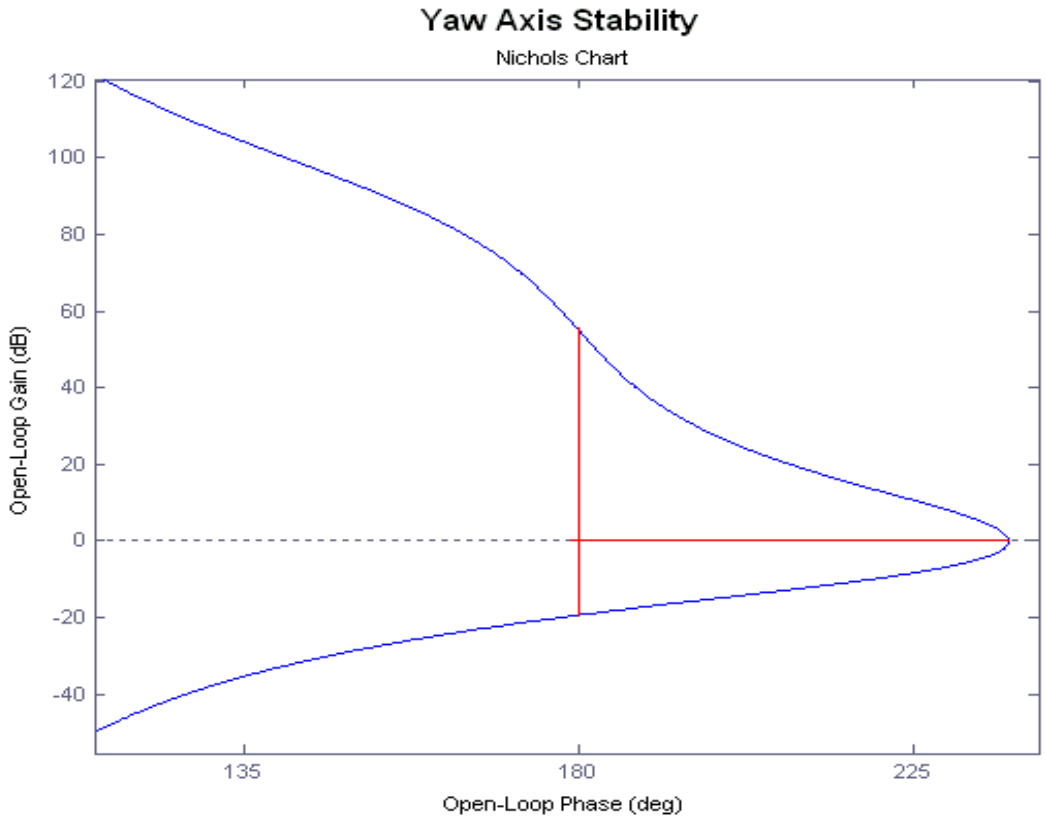


Figure 3.4.1.5 Stability margins in yaw, roll and pitch are almost identical

The closed-loop model "Closed_Loop.mdl" in Figure (3.4.1.6) performs sensitivity analysis to external disturbances in the frequency domain. Figure (3.4.1.7) shows the spacecraft attitude sensitivity response to disturbance torques. These frequency analysis models use linearized versions of the attitude control system with the PID integrator turned on. They also use the Matlab functions "Lin_ACS.m", and "Lin-Steering.m" that have all the non-linear controls taken out.

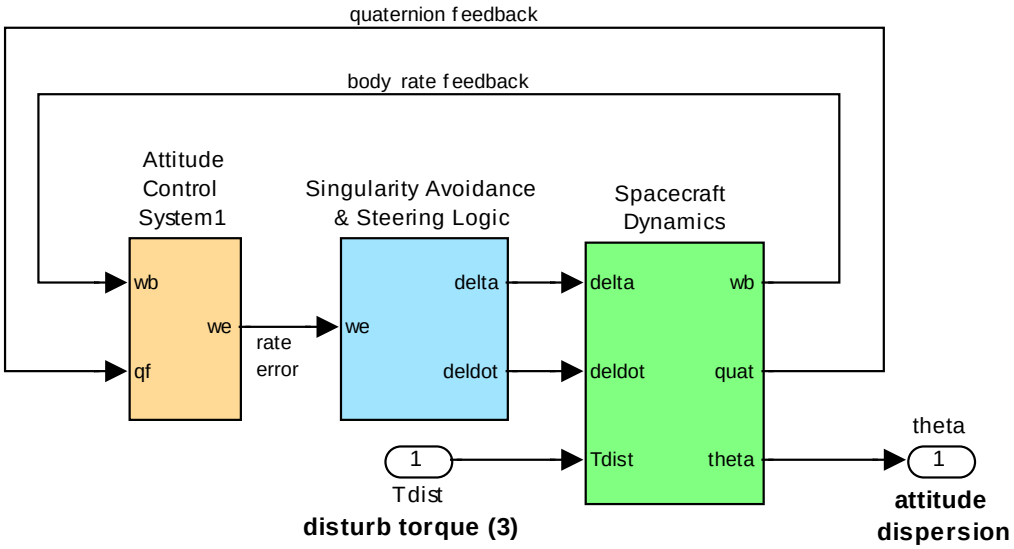


Figure 3.4.1.6 Closed-Loop Model "Closed_Loop.mdl" Used for Sensitivity Analysis

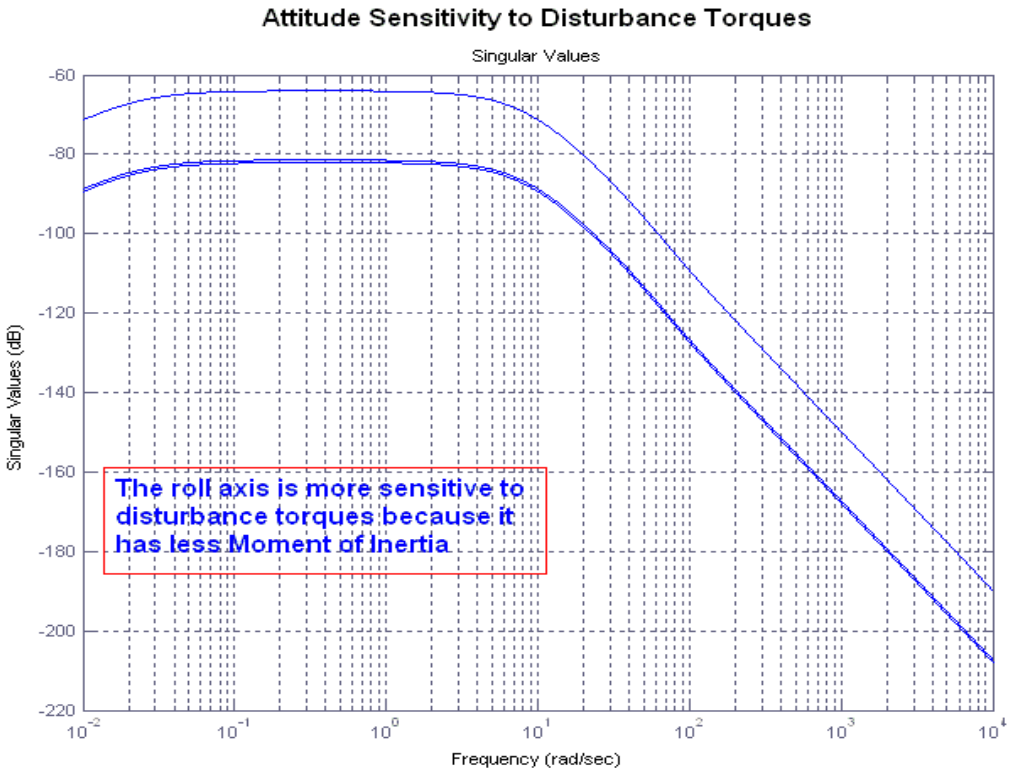


Figure 3.4.1.7 Sensitivity Analysis Sigma plot between disturbance torque (vector) and attitude errors

3.4.2 Detailed Simulation Model using 4 CMG and Max-Energy Control

Our previous CMG simulation model was a simple introduction. In this section we will upgrade it by including more details in the CMG dynamics. We previously assumed ideal CMG actuators, where the gimbal rates are equal to the commanded rates. In reality, the CMG gimbal rates are controlled by closed-loop motor driven servo systems which supply the torques that control the rates. The current simulation uses equations (3.11) and (3.12), where the CMG torque consists of two terms: the control torque T_{con} and an estimate of the gyroscopic torque ($\omega \times H_{cmg}$) which cancels out the gyroscopic effects. The CMG momentum is calculated by integrating the second part of equation (3.11). The CMG torques are calculated from equations (3.2), (3.4) and (3.10). The gimbal rates are obtained by integrating equation (3.16). The quaternion output is obtained by integrating equation (3.17).

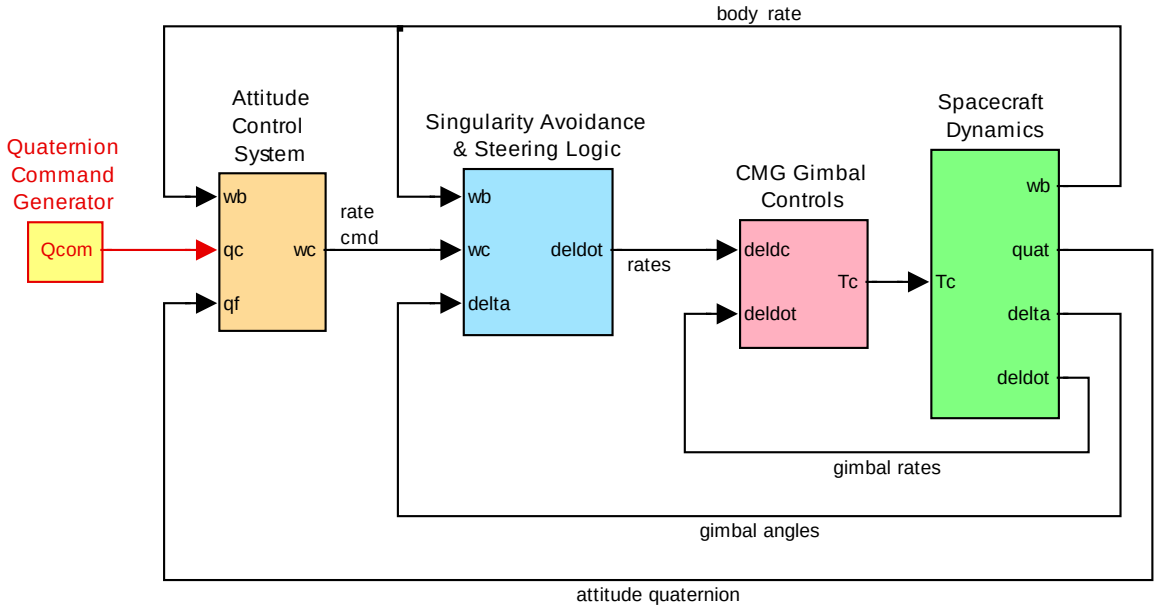


Figure 3.4.2.1 Simulation model "MaxEn_4SGCMG_Gimbals.mdl", consisting of rigid spacecraft and 4 SG-CMGs, controlled by Max-Energy ACS controller and 4 gimbal torque motor servos

The simulation model used in this example is "*MaxEn_4SGCMG_Gimbals.mdl*", shown in Figure 3.4.2.1, and it is located in folder "… *\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(b) RigBody 4SGCMG ACS w Gimbal*". It consists of four main blocks. The first block on the left hand side is a quaternion attitude controller that receives a quaternion command and the spacecraft attitude quaternion and calculates the vehicle rate command. It is implemented in Matlab function "*Max_Energy_ACS.m*" and it is identical to the ACS described in Figure 3.4.1.1. The steering logic receives the body rate command and calculates the CMG gimbal rate commands, which eventually produce the required torques on the spacecraft. It is implemented in Matlab function "*Steering.m*" and it is similar to the logic described in the previous example. The gimbal rate commands drive the gimbal motors which produce the gimbal torques required to rotate the CMG gimbals. The fourth block on the right is the spacecraft plus CMGs model which includes also the gimbal dynamics.

Figure 3.4.2.2 shows a simple diagram of the CMG gimbal control system which is used to control the rates of the CMG gimbals. It consists of a motor controlled by a PID controller. There are four of these controllers, one for each gimbal. The inputs are gimbal rate errors and the outputs are torques that drive the gimbals which are inside the green Spacecraft/ CMG dynamics block. Strong motors are sometimes needed because the control torques which rotate the gimbals have to work against strong gyroscopic torques which act like disturbances. The PID gains are calculated in the initialization script "start.m".

$$K_i = J_g \lambda \omega_{cmg}^3 \qquad K_a = (1 + 2\zeta\lambda) J_g \omega_{cmg}^2 \qquad K_b = \frac{(2\zeta + \lambda)\omega_{cmg}}{K_a}$$

where:

Jg is the CMG rotor inertia about the gimbal (slug-ft^2)
ω_{cmg} is the gimbal servo bandwidth (rad/sec)

The gains satisfy the following 3rd order characteristic equation

$$(s + \lambda\omega_{cmg})(s^2 + 2\zeta\omega_{cmg} + \omega_{cmg}^2)$$

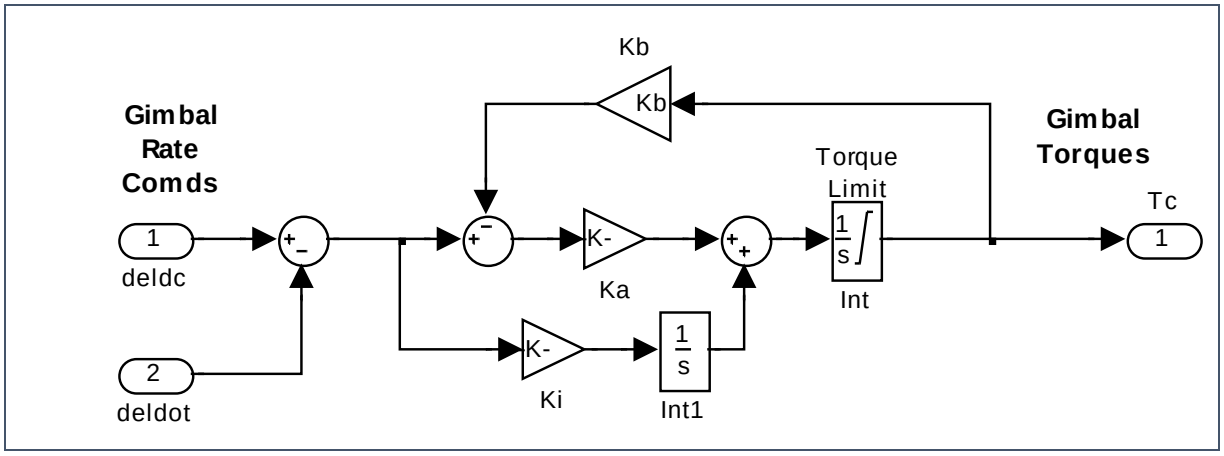


Figure 3.4.2.2 Gimbal Torque Motor with PID Control System

The spacecraft plus CMG dynamics block is shown in Figure 3.4.2.3. The inputs are CMG gimbal torques which control the gimbal rates. The outputs are body rates, attitude quaternion, gimbal angles, and gimbal rates. The dynamics are implemented in Matlab function "*SC_CMG_Gimbal.m*", shown below.

Including Gimbal Dynamics

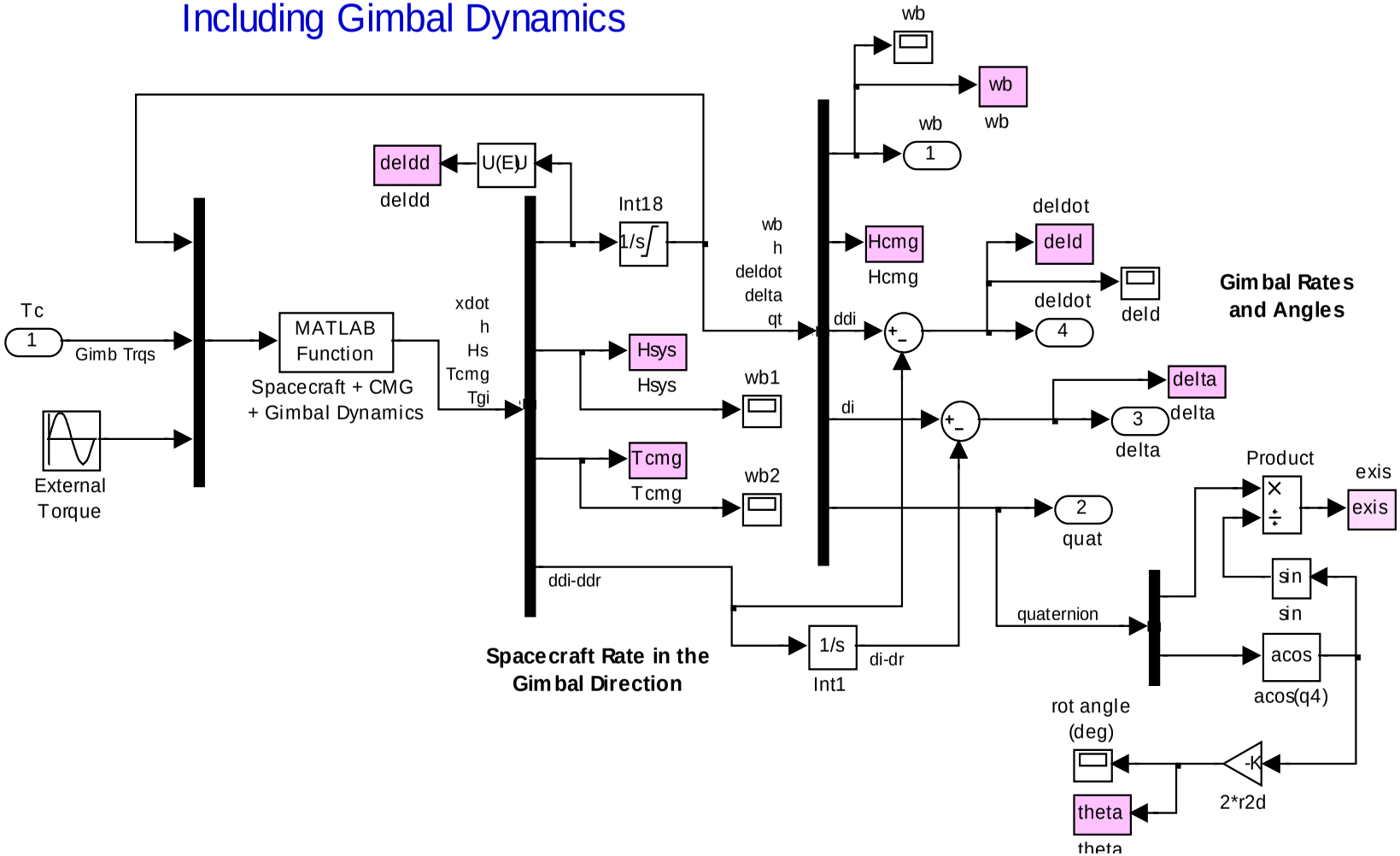


Figure 3.4.2.3 Spacecraft plus CMG Dynamics Block

Spacecraft Dynamics Function SC_CMG_Gimbal

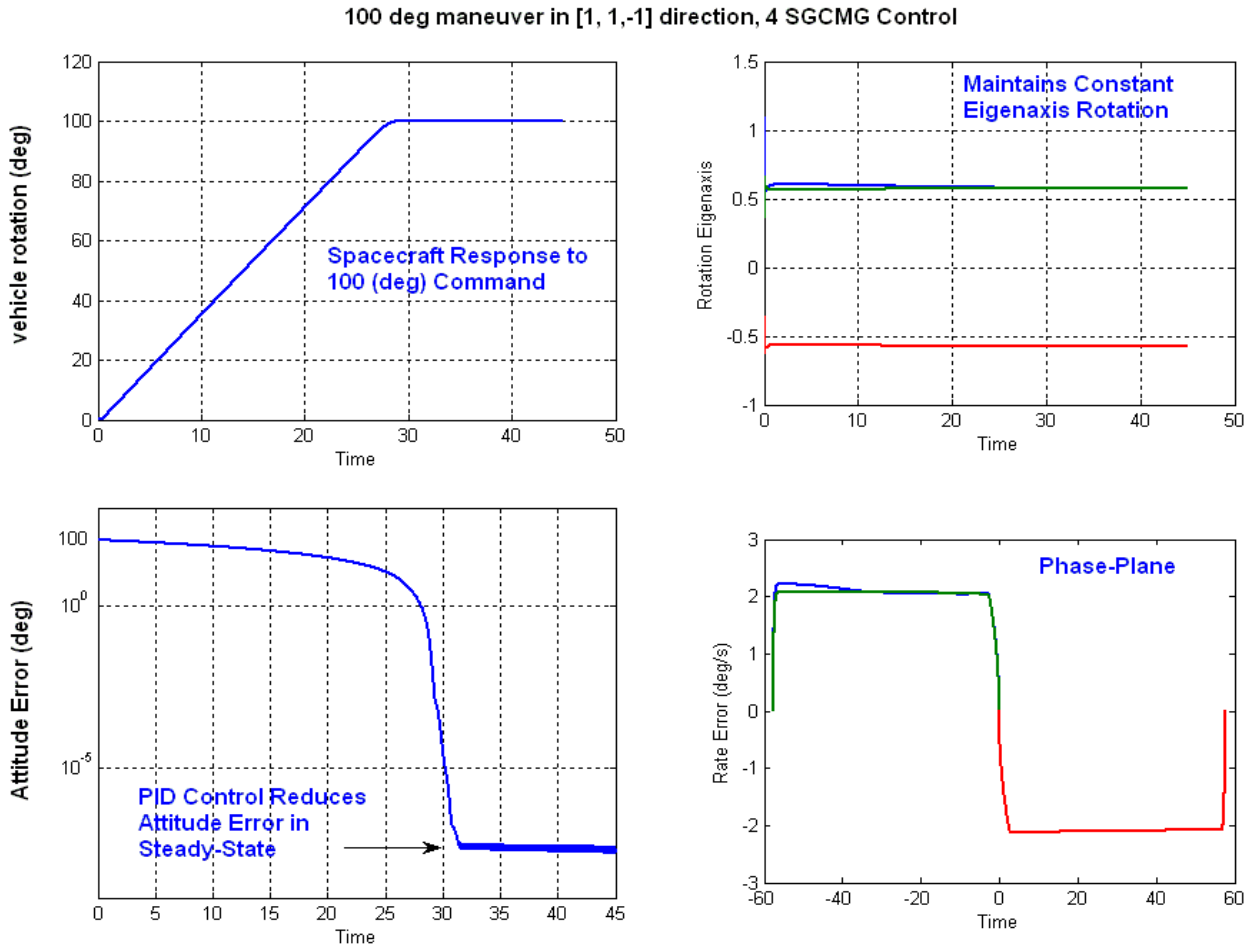
```
function xdot= SC_CMG_Gimbal(x,Tci,Td)                % s/c Dynamics with CMG
global J Jinv Jgi m ref quad hcmg d2r r2d
global bet gam Tc2b

% State Variables (x)
% x(1:3)   = Body rates   (w)        (rad/sec)
% x(4:6)   = CMG Momentum body (h)   (ft-lb-sec)
% x(7:10)  = Gimbal rates (delt-dot) (rad/sec)
% x(11:14) = Gimbal angles (delta)   (rad)
% x(15:18) = Quaternion
% Inputs:
% Tci(4)   = Gimbal Torques (ft-lb)
% Td(3)    = Disturbance Torque (ft-lb)

xdot= zeros(28,1);
w =     x(1:3);                                       % Body rates
h =     x(4:6);                                       % CMG Momentum
deldot= x(7:10);                                      % Gimbal rates
delta = x(11:14);                                     % Gimbal Angles
qt=     x(15:18);                                     % Quaternion
[Pj,thd,phd,ddd]= Transforms(bet,gam,delta,w);

Tcmg=zeros(3,1); h2=Tcmg;
for i=1:4
    Mj=[Tci(i); ...                                   % CMG Torque in CMG axis
        hcmg(i)*deldot(i); ...
        0];
    Tcmg= Tcmg -Tc2b*Pj(:,:,i)*Mj;                    % Torque on Vehicle
%   h2= h2 + (cos(delta(i))*ref(:,i) + sin(delta(i))*quad(:,i))*hcmg(i);
end

Hs = J*w + h;                                         % System Momentum

xdot(1:3)= Jinv*(Tcmg -cross(w,J*w) +Td);             % Vehicle acceleration
wdot=xdot(1:3);                                       % Vehicle Acceleration
xdot(4:6)=-Tcmg -cross(w,h);                          % H-dot
for i=1:4
    sd=sin(delta(i)); cd=cos(delta(i));
    xdot(6+i)= (Tci(i)-hcmg(i)*(thd(i)*sd-phd(i)*cd))/Jgi;  % Gimbal accelr delt-ddot
    xdot(10+i)= x(6+i);                               % Gimbal rates  delta-dot
end

xdot(15:18)= 0.5*[0    w(3)  -w(2)   w(1);            % Quaternion Update
              -w(3)    0      w(1)   w(2);
               w(2)   -w(1)   0      w(3);
              -w(1)   -w(2)  -w(3)   0]*qt;
xdot(19:21)= Hs;                                      % System Momentum Hs
xdot(22:24)= Tcmg;                                    % CMG Torques on Vehi
xdot(25:28)= ddd;                                     % delta_dot (Inert-Relat)
```

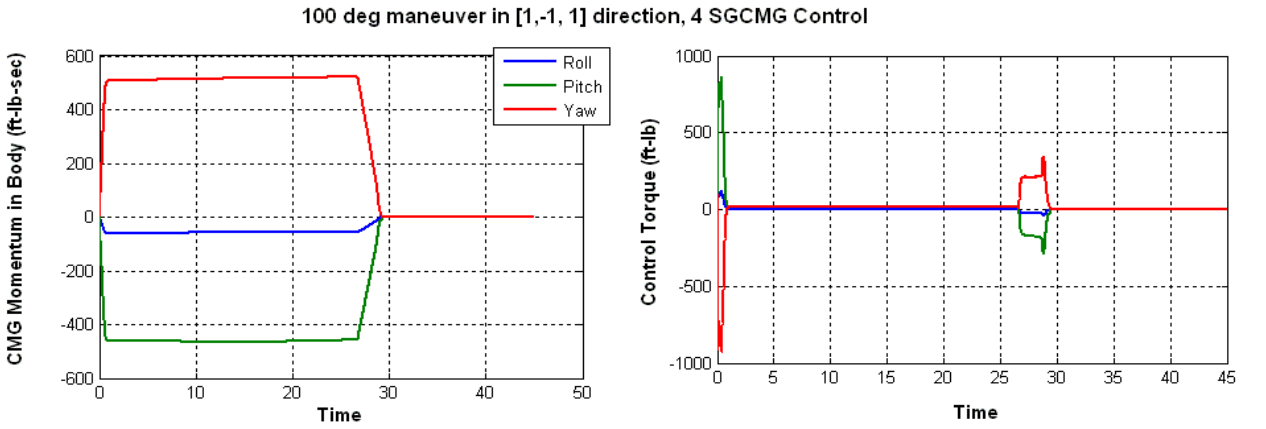
Simulation Results

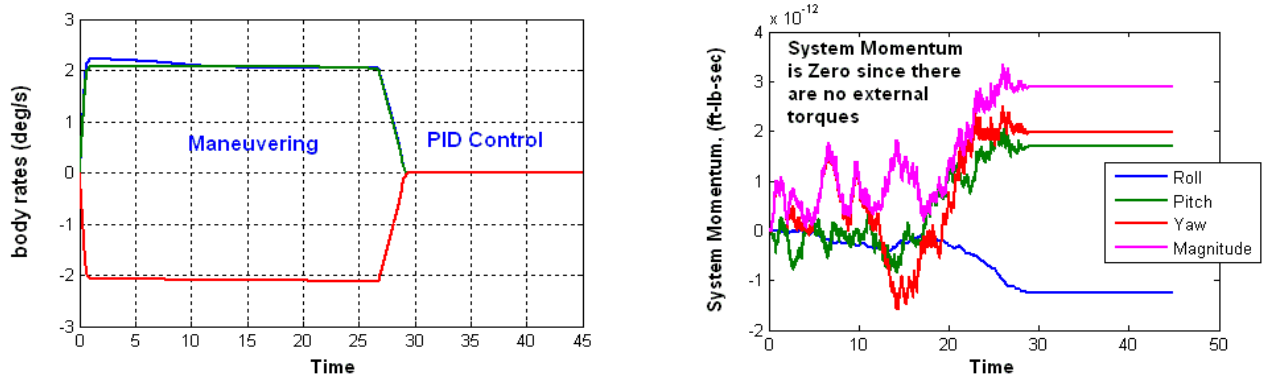
The Simulink model described will be used here to demonstrate a couple of maneuvering simulations. The simulation and maneuver parameters are loaded into Matlab by the initialization file "start.m". In the first case we have a typical maneuver without singularity problems. In the second case a singularity occurs while maneuvering. The file "pl.m" plots the simulation results.

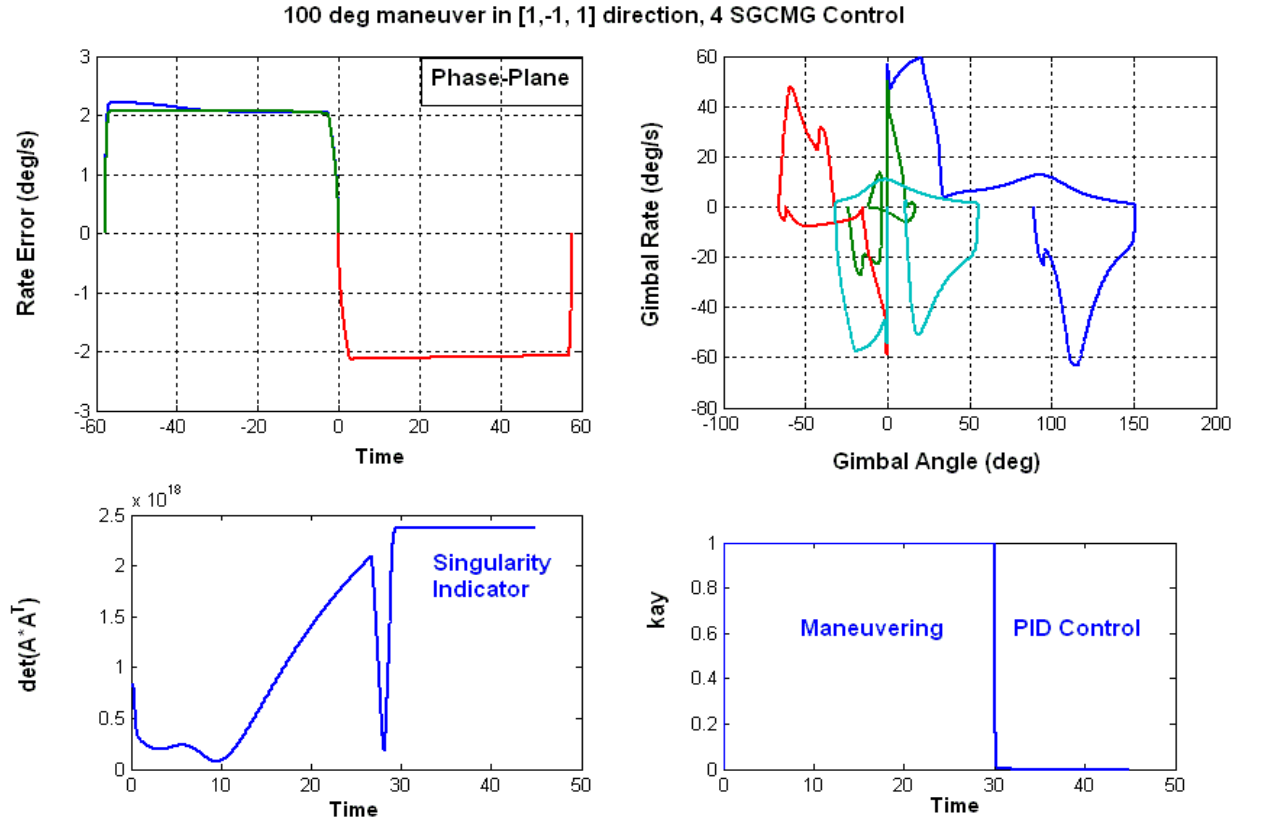
Simulation 1 A typical 100 (degrees) maneuver (without singularity)

In this case we are commanding the spacecraft to rotate 100º in an arbitrary direction (1, 1, -1). The algorithm uses a singularity proximity measure, which is the determinant of the AA^T matrix. When this measure becomes very small it is an indication of singularity occurrence. In this case we hit a low point at 10 seconds but it is not a real singularity. The spacecraft maneuvers smoothly towards the commanded direction maintaining an almost constant eigenaxis, body rate, and CMG momentum. The CMG torque and phase-plane show an acceleration pulse in the beginning and at the end of the maneuver. The deceleration is not as high and lasts longer than the acceleration. Shortly before the 30 seconds when the error becomes small the "kay" factor drops to zero which turns on the PID controller that keeps the steady-state error in the 10^{-7} level. The four gimbal rates versus gimbal angles are also shown in phase-plane. Notice that the system momentum is maintained at zero throughout the maneuver.



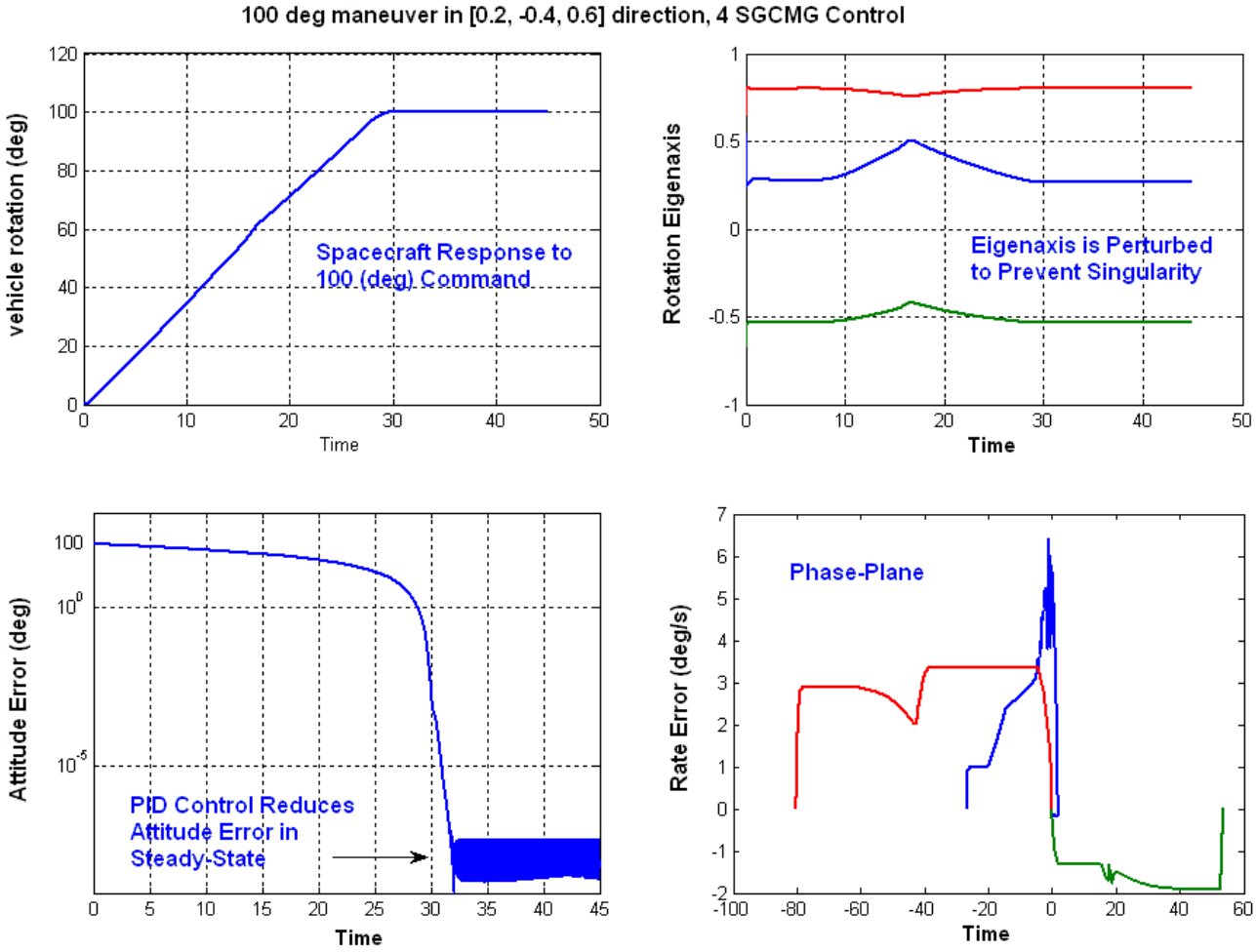

100 deg maneuver in [1,-1, 1] direction, 4 SGCMG Control
CMG Momentum in Body (ft-lb-sec)
Time
Roll
Pitch
Yaw
Control Torque (ft-lb)
Time

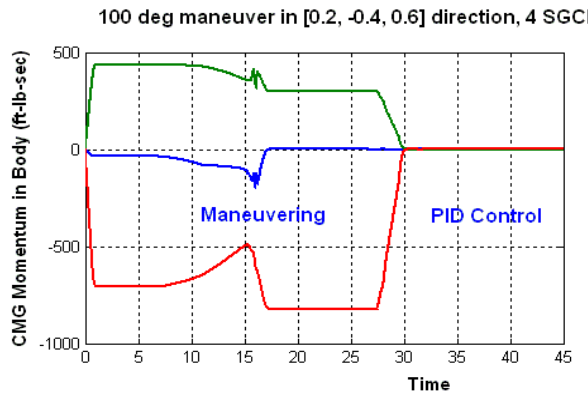

body rates (deg/s)
Maneuvering
PID Control
Time
x 10^-12
System Momentum is Zero since there are no external torques
System Momentum, (ft-lb-sec)
Roll
Pitch
Yaw
Magnitude
Time

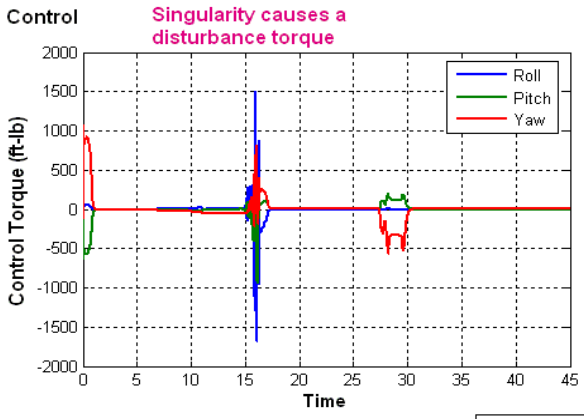

100 deg maneuver in [1,-1, 1] direction, 4 SGCMG Control
Rate Error (deg/s)
Phase-Plane
Time
Gimbal Rate (deg/s)
Gimbal Angle (deg)
x 10^18
det(A*A^T)
Singularity Indicator
Time
kay
Maneuvering
PID Control
Time

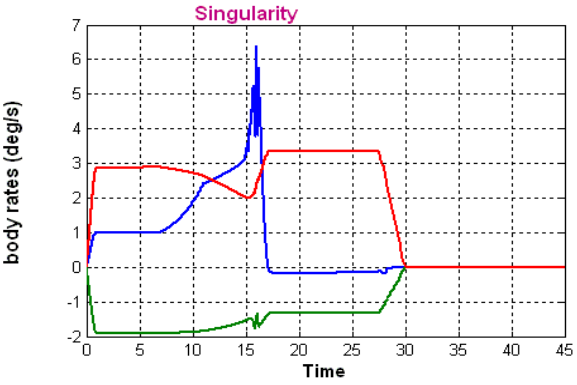
Simulation 2 100 (degrees) maneuver with Singularity Present

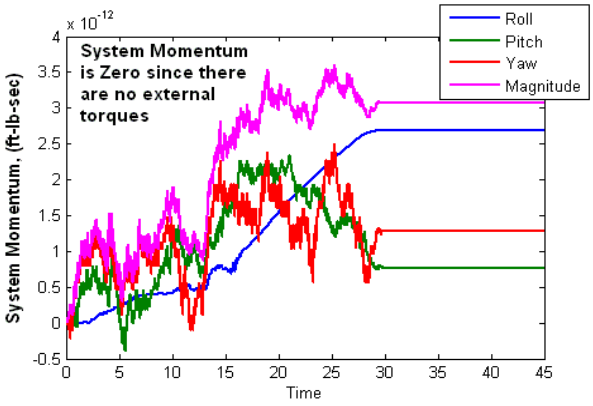
The following simulation results demonstrate the effectiveness of the singularity avoidance algorithm to prevent a gimbal lock situation. Gimbal lock occurs when the CMGs cannot provide the required torque in the direction demanded by the attitude control logic. The algorithm uses a singularity proximity measure, which is the determinant of the AA^T matrix, and when this measure becomes very small it introduces a perturbation at the gimbals to get around the singularity. In the following simulation we are commanding a 100º maneuver in the direction (0.2, -0.4, 0.6). At approximately 7 seconds the determinant of the singularity measure drops to almost zero. A perturbation signal is introduced by the singularity avoidance algorithm that distorts the constant body rates and CMG momentum which were observed in the previous simulation case during maneuvering. It also causes a CMG torque disturbance at 16 seconds. Despite the singularity occurrence the spacecraft attitude continues to rotate towards the target direction, not directly as in the previous simulation where the eigenaxis was maintained constant, but it converges towards the target and at approx 30 seconds when the error becomes small the "kay" factor drops to zero which turns on the PID controller that keeps the steady-state error in the 10^{-7} level. Notice that the system momentum is maintained at zero throughout the maneuver.

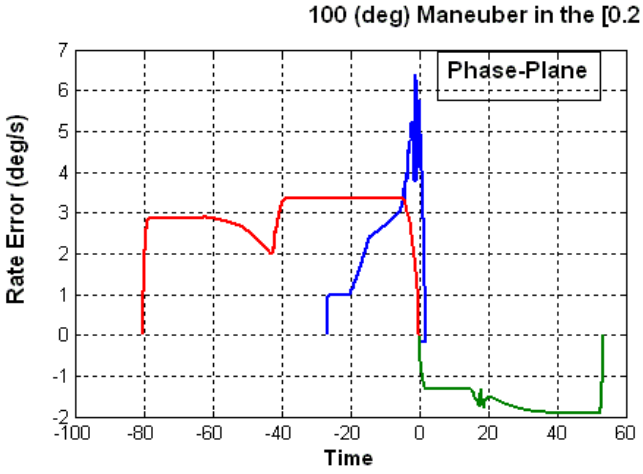


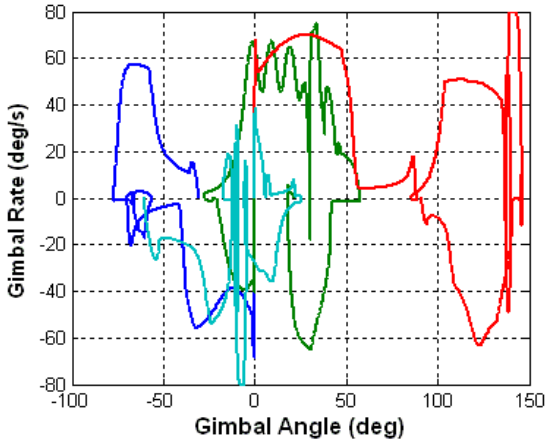

100 deg maneuver in [0.2, -0.4, 0.6] direction, 4 SGCMG Control
CMG Momentum in Body (ft-lb-sec)
Maneuvering
PID Control
Time

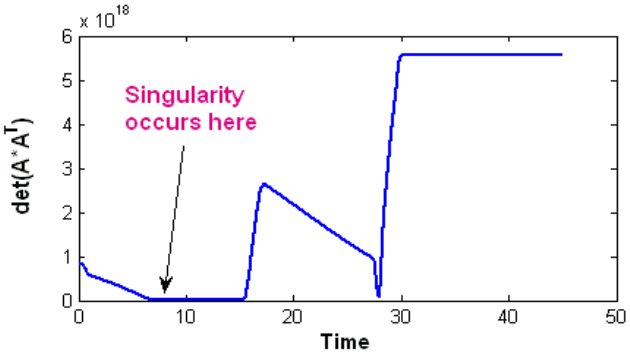

Singularity causes a disturbance torque
Control Torque (ft-lb)
Roll
Pitch
Yaw
Time

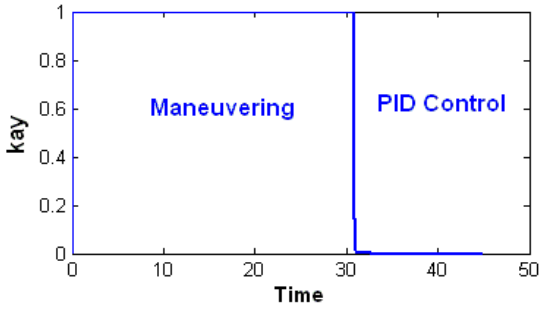

Singularity
body rates (deg/s)
Time


System Momentum is Zero since there are no external torques
System Momentum, (ft-lb-sec)
Roll
Pitch
Yaw
Magnitude
Time


100 (deg) Maneuber in the [0.2 -0.4 0.6] direction with Singularity
Phase-Plane
Rate Error (deg/s)
Time


Gimbal Rate (deg/s)
Gimbal Angle (deg)


Singularity occurs here
det(A*A^T)
Time


Maneuvering
PID Control
kay
Time

3.4.3 Adding Flexibility to the Four CMG Simulation Model

This time we will go one step further and include structural flexibility. The Simulink model in this example is "*MaxEn_4SGCMG_Gimbals_Flex.mdl*", shown in Figure (3.4.3.1). The Matlab and other data files used are in directory "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(c) Flex 4SGCMG ACS w Gimbal*". It is very similar to the previous model, described in section 3.4.2, but it includes structural bending. The attitude control law and steering logic are the same, but some of the control parameters were adjusted to accommodate flexibility. The simulation data are loaded into Matlab by executing file "Start.m", and the file "pl.m" plots the simulation results, as in previous examples.

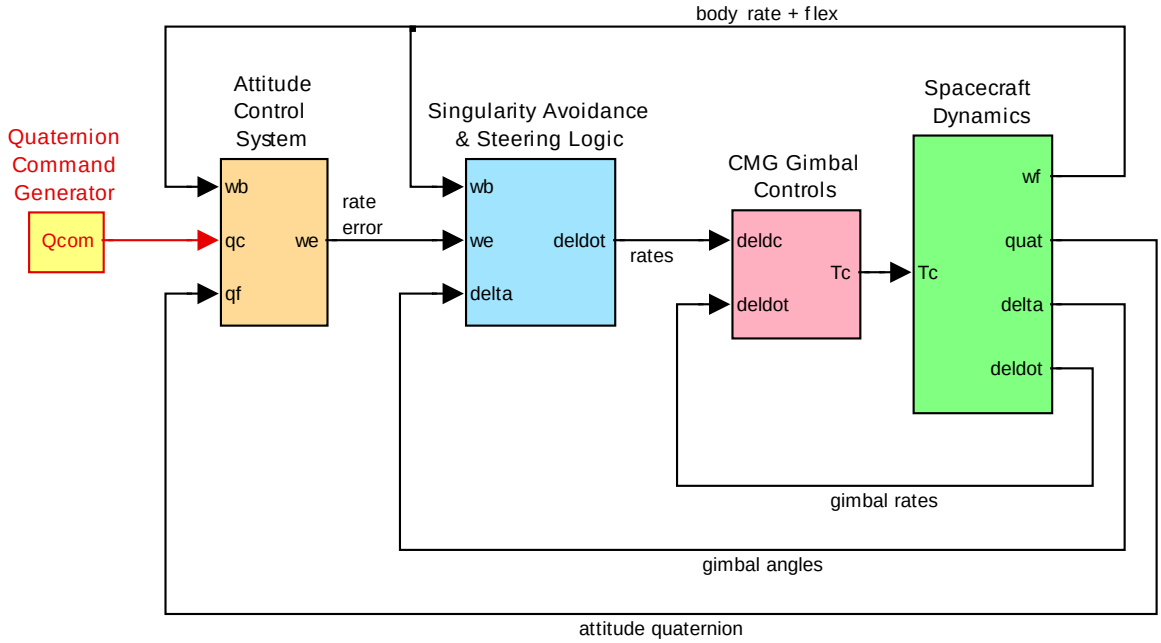


Figure 3.4.3.1 Simulation model "MaxEn_4SGCMG_Gimbals.mdl", consisting of rigid plus flexible spacecraft dynamics and 4 SG-CMGs

As you can see in Figure 3.4.3.2, the main difference of this simulation model from the previous one shown in figure 3.4.2.3 is that the spacecraft dynamics (green) block includes flexibility. The structural flexibility subsystem is a state-space model shown in detail in Figure 3.4.3.3. It is loaded from file "*flex_only_fem_s.m*" which was created by Flixan "flex spacecraft modeling program" in section 1.1 and its title is "*Flex Only Spacecraft with RCS and CMG*". It has the rigid-body modes removed because the rigid-body dynamics is implemented in Matlab function "*SC_CMG_Gimbal.m*", which contains also the CMG dynamics. The coupling between the rigid and flex models is not straightforward. The 4 CMG gimbal torques drive the rigid-body dynamics as they are controlled by the 4 gimbal servos attempting to maintain the commanded gimbal rates. The torque motors have to be strong enough because in addition to the CMG load inertia they also have to fight the gyroscopic torque $h_{cmg}\left(\dot{\theta}\sin\delta - \dot{\phi}\cos\delta\right)$ which is not small.

The inertial gimbal rates $\dot{\delta}_i$ rotate the CMG momentum vectors generating torques which produce body rates, and attitudes. The combined CMG torque in vehicle axes (T_{cmg}) is also driving the

flexibility state-space model to excite the structure modes. The flex model output consists of the flex contributions of the body rates which are added to the rigid-body rates to produce the total body rates. The quaternion is updated using the total body rates, including flexibility. The function "SC_CMG_Gimbal.m" which implements the spacecraft dynamics is listed below.

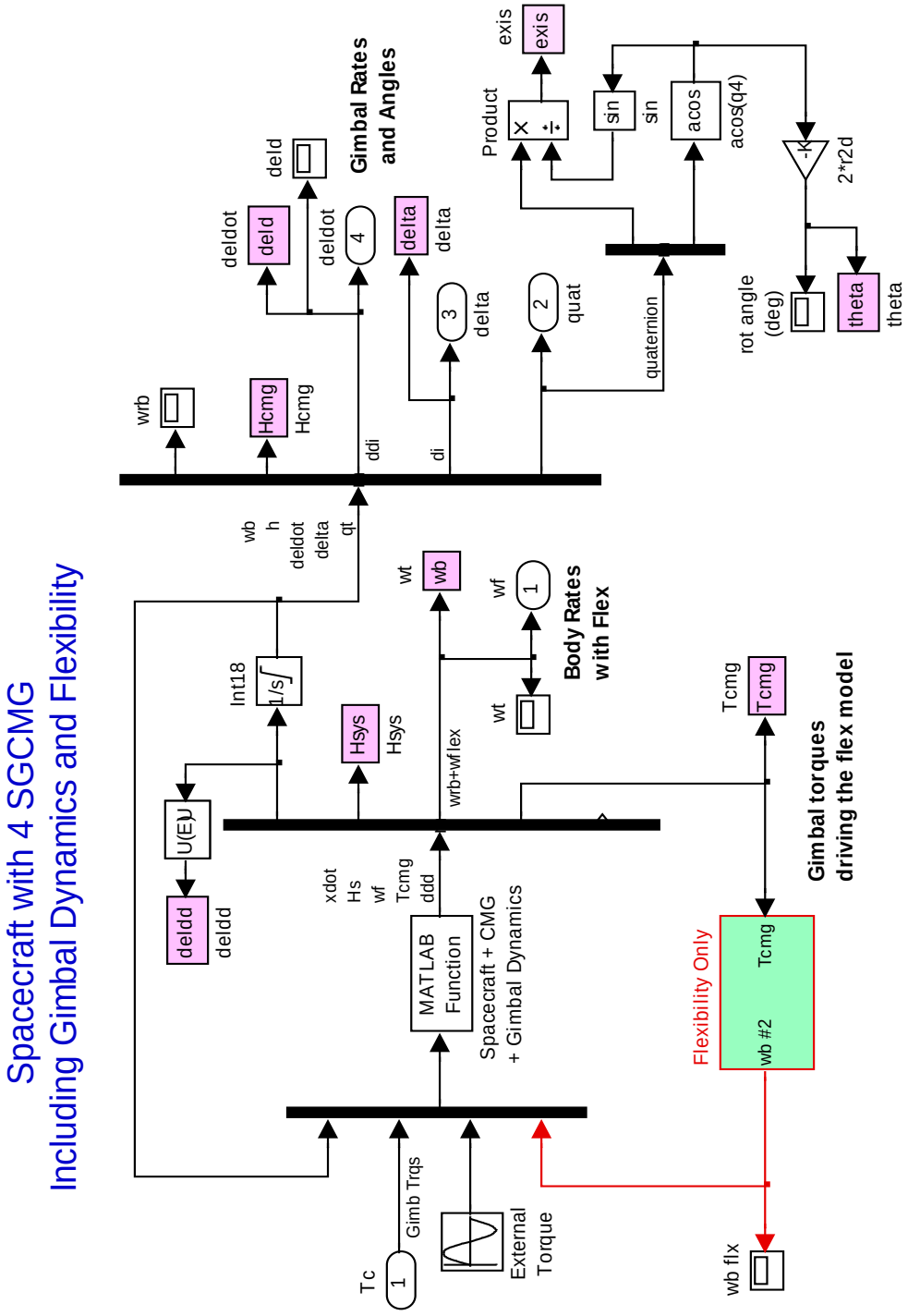


Figure 3.4.3.2 Spacecraft dynamics subsystem includes CMG gimbals plus structural flexibility

Spacecraft Dynamics Function SC_CMG_Gimbal.m

```
function xdot= SC_CMG_Gimbal(x,Tci,Td,wf)        % s/c Dynamics with CMG
global J Jinv Jgi m ref quad hcmg d2r r2d
global bet gam Tc2b Jgi Jsi Joi

% State Variables (x)
% x(1:3)   = Body rates   (w)       (rad/sec)
% x(4:6)   = CMG Momentum body (h)   (ft-lb-sec)
% x(7:10)  = Gimbal rates (delt-dot) (rad/sec)
% x(11:14) = Gimbal angles (delta)   (rad)
% x(15:18) = Quaternion
% Inputs:
% Tci(4)   = Gimbal Torques (ft-lb)
% Td(3)    = Disturbance Torque (ft-lb)
% wf(3)    = Flex Body Rate

xdot= zeros(31,1);
w =     x(1:3);                                  % Rigid-Body rates
wt=     w+wf;                                    % Total body rate
h =     x(4:6);                                  % CMG Momentum
deldot= x(7:10);                                 % Gimbal rates
delta = x(11:14);                                % Gimbal Angles
qt=     x(15:18);                                % Quaternion
[Pj,thd,phd,ddd]= Transforms(bet,gam,delta,w);

Tcmg=zeros(3,1); h2=Tcmg;
for i=1:4
   sd=sin(delta(i)); cd=cos(delta(i));
   Mj=[Tci(i); ...                                % CMG Torque in CMG axis
      deldot(i)*((Jsi-Jgi)*(thd(i)*cd+phd(i)*sd) +hcmg(i)); ...
      deldot(i)* (Jgi-Joi)*(phd(i)*cd-thd(i)*sd)];
   Tcmg= Tcmg -Tc2b*Pj(:,:,i)*Mj;                 % Torque on Vehicle
%   h2= h2 + (cos(delta(i))*ref(:,i) + sin(delta(i))*quad(:,i))*hcmg(i);
end

Hs = J*w + h;                                    % System Momentum

xdot(1:3)= Jinv*(Tcmg -cross(w,J*w) +Td);        % Vehicle acceleration
wdot=xdot(1:3);                                  % Vehicle Acceleration
xdot(4:6)=-Tcmg -cross(w,h);                     % H-dot
for i=1:4
   sd=sin(delta(i)); cd=cos(delta(i));
   xdot(6+i)= (Tci(i)-hcmg(i)*(thd(i)*sd-phd(i)*cd))/Jgi;  % Gimbal accelr delt-ddot
   xdot(10+i)= x(6+i);                           % Gimbal rates  delta-dot
end

xdot(15:18)= 0.5*[0    wt(3)  -wt(2)    wt(1);        % Quaternion Update
                  -wt(3)   0     wt(1)   wt(2);
                   wt(2)  -wt(1)  0       wt(3);
                  -wt(1)  -wt(2)  -wt(3)  0]*qt;
xdot(19:21)= Hs;                                 % System Momentum Hs
xdot(22:24)= wt;                                 % Total Vehi rate
xdot(25:27)= Tcmg;                               % CMG Torques on Vehi
xdot(28:31)= ddd;                                % delta_dot (Inertial-Relativ)
```

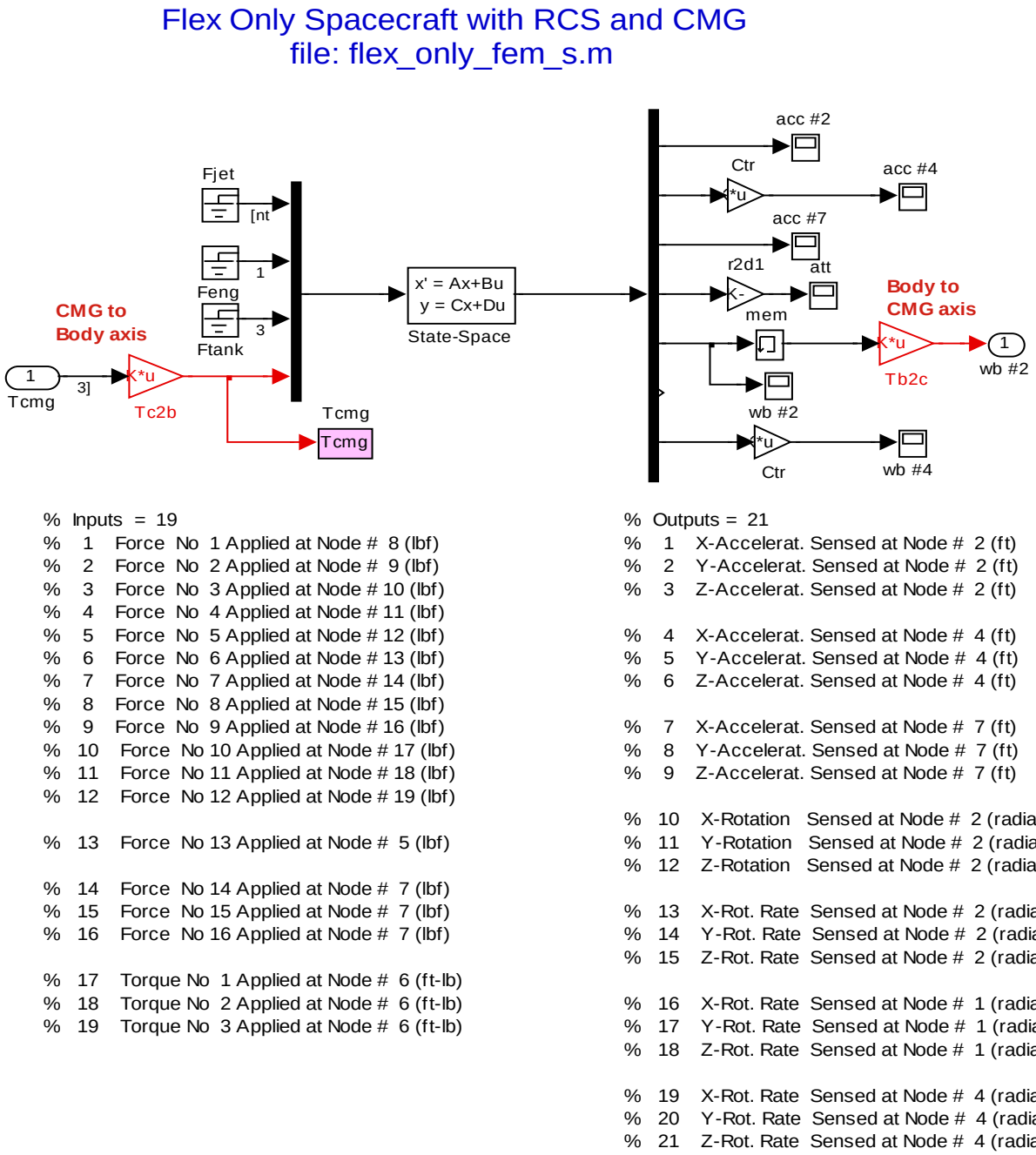


Figure 3.4.3.3 Flex model is in State-Space form in file (flex_only_fem_s.m)

The transformation matrices from CMG to body axes (Tc2b) and from body to CMG (Tb2c) are needed in order to couple the flex model with the rigid-body dynamics. The flex model is in body coordinates, but the spacecraft model (originally defined in body) was transformed to CMG pyramid coordinates. So the combined model is in CMG array coordinates.

The Max-Energy attitude control law is shown in Figure 3.4.3.4. It is implemented in function "*Max_Energy_ACS.m*" and it operates in two modes: the maneuvering mode, and the PID control mode. In the beginning of the maneuver, kay=1. The integrator of the PID is turned off, and the spacecraft rate is commanded a non-linear parabolic trajectory which is a function of the quaternion attitude error. The spacecraft rate is also limited during maneuvering to a steady rate along the commanded eigenaxis. When the rate versus attitude error trajectory reaches the parabolic line in the phase-plane the trajectory converges towards the origin, bringing both the rate and attitude error to zero simultaneously in all 3 directions. When the attitude error becomes sufficiently small (dictated by the logic in the ACS) the "kay" factor switches to zero turning the PID integrators on, to further attenuate the attitude error. The switching of kay from 1 to zero is a function of both: rate and attitude error magnitudes combined. This error magnitude signal is filtered (by filter state x_6) to smooth out the transitioning of the "kay" factor from maneuvering to PID control. The states x_4 and x_5 in the ACS are also used to provide a clean transitioning of kay.

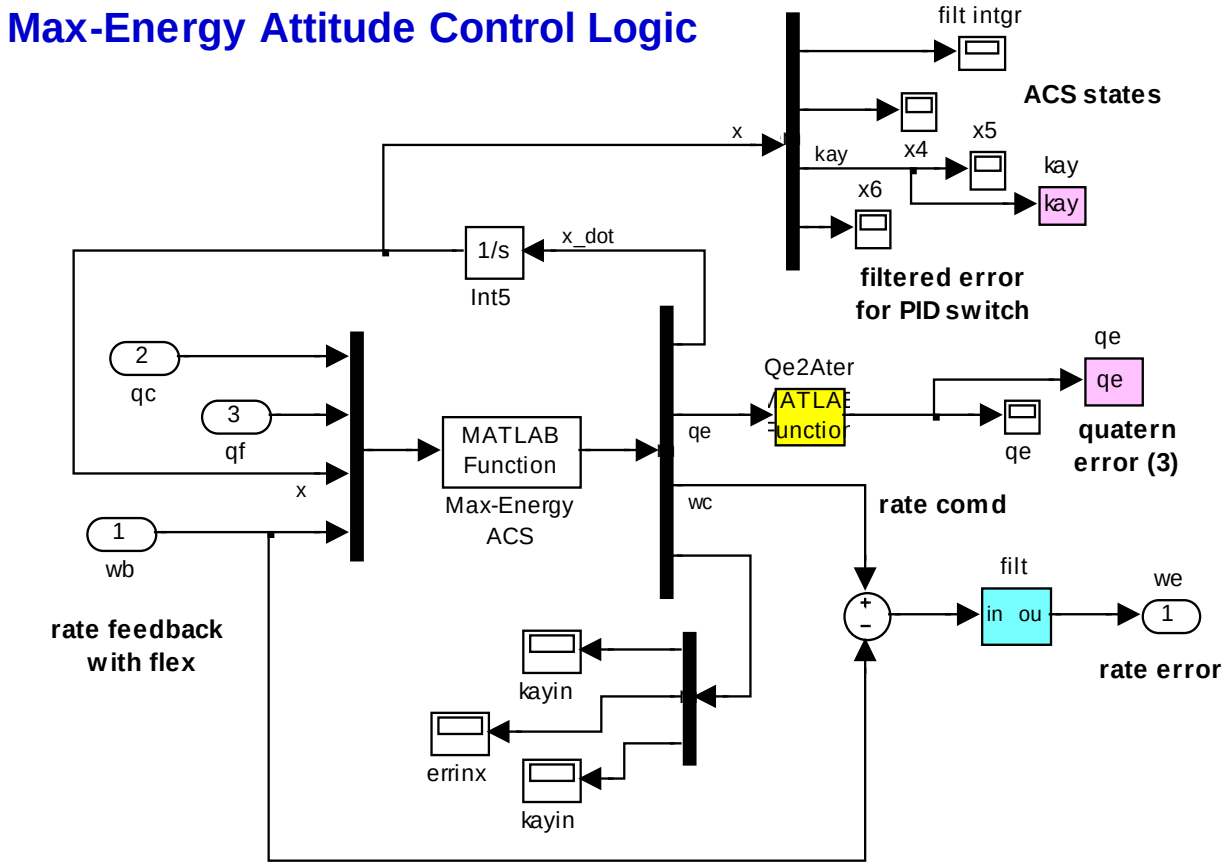


Figure 3.4.3.4 Max Energy Attitude Control Law is implemented in function (Max_Energy_ACS.m)

The CMG steering logic is shown in Figure 3.4.3.5. It controls the spacecraft rate as commanded by the ACS. It is the D part of the PID control law. It receives the spacecraft rate command from the ACS and it commands the gimbal rates as needed in order to achieve the required CMG control torques on the spacecraft. Initially it calculates an acceleration command proportional to the rate error. This acceleration command is limited, as needed, proportionally in all directions in order to maintain an eigenaxis rotation.

The algorithm calculates the matrix A(δ), as shown in equation 3.15 and uses pseudo-inverse to invert it, as shown in equation 3.3.4, in order to calculate the gimbal rate commands. Since A is a function of δ the steering function also needs the gimbal angles δ as inputs at every instant. The singularity avoidance logic is also included here. The clock input is needed for the phasing in the off-diagonal elements in the singularity avoidance perturbation matrix (E).

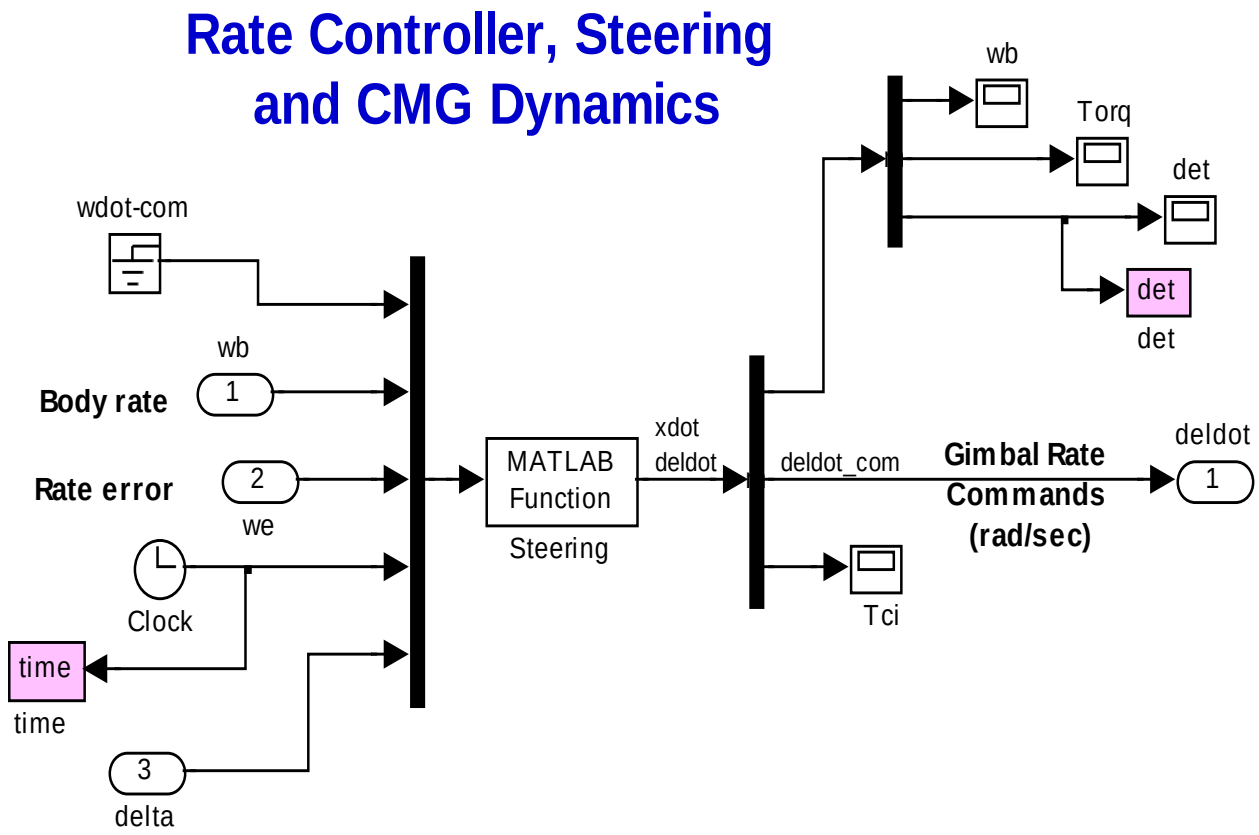


Figure 3.4.3.5 Rate Control and CMG Steering Logic, implemented in function "Steering.m"

Initialization File "Start.m"

The initialization file "start.m" initializes the simulation. It loads the vehicle mass properties in body coordinates which are transformed to CMG pyramid coordinates, see Figure(). It also loads the flexibility state-space model, the transformation matrices, CMG parameters, orientation angles, gimbal rate limits, and gains for the gimbal control system. It initializes also the spacecraft state-vector, sets torque limits, max acceleration and rate limits, and parameters for the singularity avoidance logic.

```
global J Jinv m ref quad hcmg d2r r2d wcmg zeta
global Acc_Lim Rat_Lim kr ki kp Es Ps Fs wlim mx
global bet gam Jgi Jsi Joi
d2r= pi/180; r2d=180/pi;

[Af, Bf, Cf, Df] = flex_only_fem_s;              % Load Flex Only Spacecr Dynamics
J= [0.17E+94, -0.16e+93,  0.11E+92;              % Vehicle MOI matrix in (slug-ft^2).
   -0.16e+93,  1.32E+94,  0.31E+92;
    0.11E+92,  0.31E+92,  1.41E+94];
Tc2b= [0, 0, 1; ...                              % Transformation Matrix
       0, 1, 0; ...                              % from CMG to Body Axis
      -1, 0, 0]; Tb2c= inv(Tc2b);
J= Tb2c*J*Tc2b;  Jinv= inv(J);                   % Convert Inertias to CMG axes
Ctr= Tb2c*[0 1 0; 1 0 0; 0 0 -1];                % Transform Matrix (append to body to CMG axis)

Jsi= 1.2; Jgi= 0.6; Joi= 0.8;                    % CMG Inertia about Spin, Gimbal, Outp axes
wcmg=500; zeta= 0.95; lamb=0.5;                  % CMG servo bandwidth (rad/s), damping coeff
Kii=lamb*Jgi*wcmg^3;                             % CMG Servo Gains
Ka=(1+2*zeta*lamb)*Jgi*wcmg^2;
Kb=(2*zeta+lamb)*wcmg/Ka;
hcmg=[1, 1, 1, 1]*1200;                          % CMG Momentum capability

% CMG Geometry
bet=68;                                          % Pyramid Beta angle
gam=[90, 180, 270, 0];                           % Pyramid Gamma angles
[m, ref, quad]= M4(bet);                         % CMG Gimbal, Spin, & Torq direction matrices

sigma0=[0; 0; 0; 0]*d2r;                         % Initial CMG Gimbal positions (rad)
wb0= [0; 0; 0];                                  % Initial body rates
h0= [0; 0; 0];                                   % Initial CMG Momentum (body)
deldot= [0 0 0 0]';                              % Initial Gimb Rates
delta = [0 0 0 0]';                              % Initial Gimb Angles
Qt0=[0; 0; 0; 1];                                % Initial Quaternion
ini= [wb0',h0',deldot',delta',Qt0'];             % State Integrator Initialization
wlim= 3.6*d2r;                                   % MaxEn ACS Rate Limit 3.9 (deg/s)
Tmax= 650;                                       % Max CMG Torque 650
mx= Tmax*[1,1,1]./[J(3,3),J(2,2),J(3,3)];        % Max accelerations x,y,z
Acc_Lim=30*d2r; Rat_Lim=12*d2r;                  % Steer Law Accel & Rate limit (deg/sec)
Tglim=120; Wclim=100*d2r;                        % CMG Model Gimbal Accelerat and Rate Limits
ki=0.07; kp=1.4; kr=10;                          % PID Gains: ki=0.3; kp=7.3; kr=12
Es=0.05; Ps=1.0e18; Fs=0.2;                      % Singul Avoid Param Es=0.01 Ps=1.0e11 Fs=0.2
xlim= inf(1,18); xlim(7:10)=Wclim*[1 1 1 1];     % State Integrator Limits

com_dir=Tb2c*[-1; 1; 1];                         % Command Direct Unit Vector (body)
com_dir=com_dir/sqrt(com_dir'*com_dir);          % Unit Vector [0.7; -0.6; 0.4]
```

Maneuvering Simulations Using the Non-Linear Flex Model

The following three simulation results were obtained using the non-linear Simulink model "*MaxEn_4SGCMG_Gimbals_Flex.mdl*".

Case # 1, an 80° Maneuver in direction: (-1, 1, 1)

In the first simulation, Figure (3.4.3.6), the spacecraft is commanded to maneuver 80° in direction: (-1, 1, 1). It performs an almost perfect eigenaxis rotation maintaining nearly constant rates during maneuvering. The 0.5 Hz structural mode affects mainly the roll axis, but it does not degrade the flex stability. The maneuver is very symmetrical as shown in the attitude phase-plane. The CMG control torque shows a big torque spike which gets the maneuver started. There is an almost zero torque while maneuvering, and a decelerating torque pulse to stop the rotation. The decelerating pulse is not as strong as the initial torque pulse. Although the rates are the same in all 3 axes the CMG momentum is different in the 3 directions because the spacecraft moments of inertia are different. The singularity indicator shows an initial low value in the determinant (close to singularity) but it does not last very long and it climbs to higher values later in the maneuver. The attitude error plot shows when it is in PID control the integrator gradually reduces the attitude error to very low values. In the absence of external disturbances the total system momentum remains zero during the maneuver.

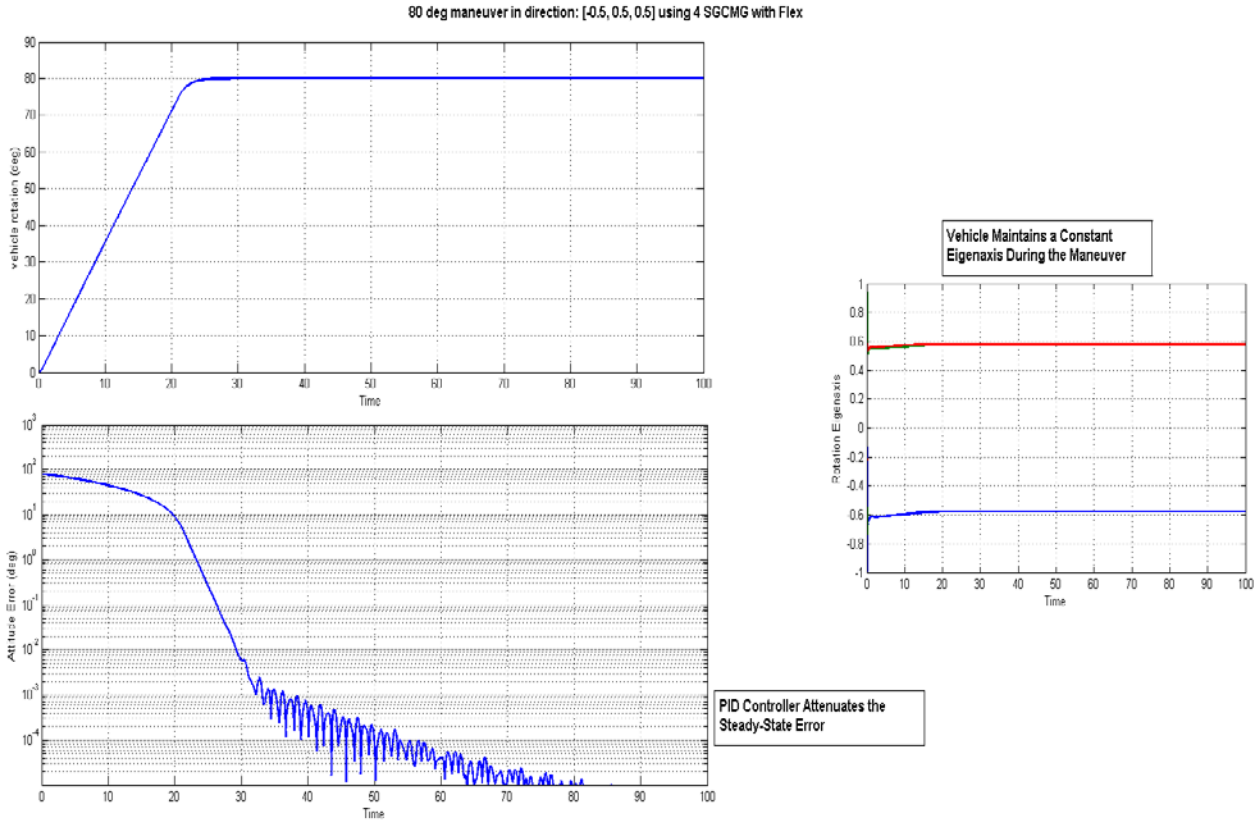
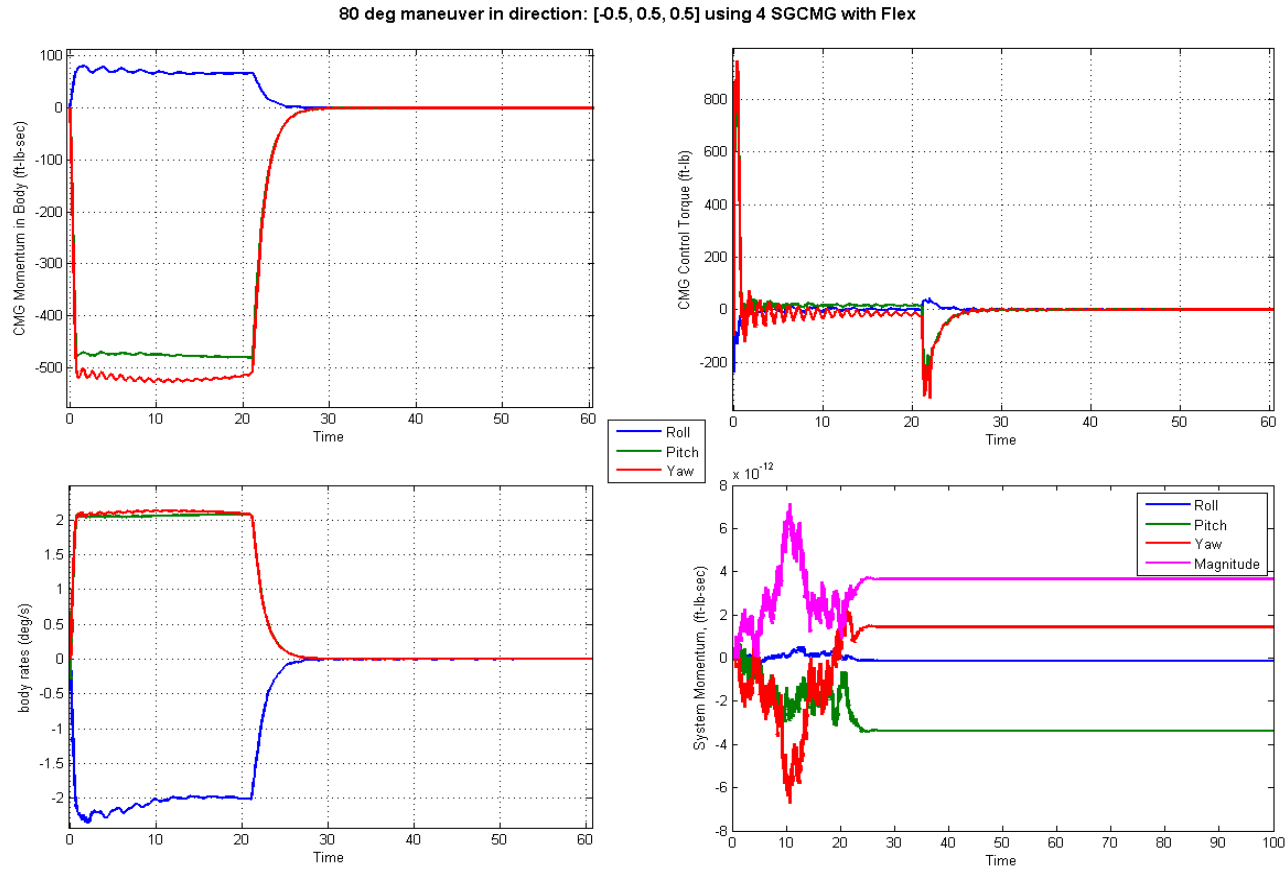
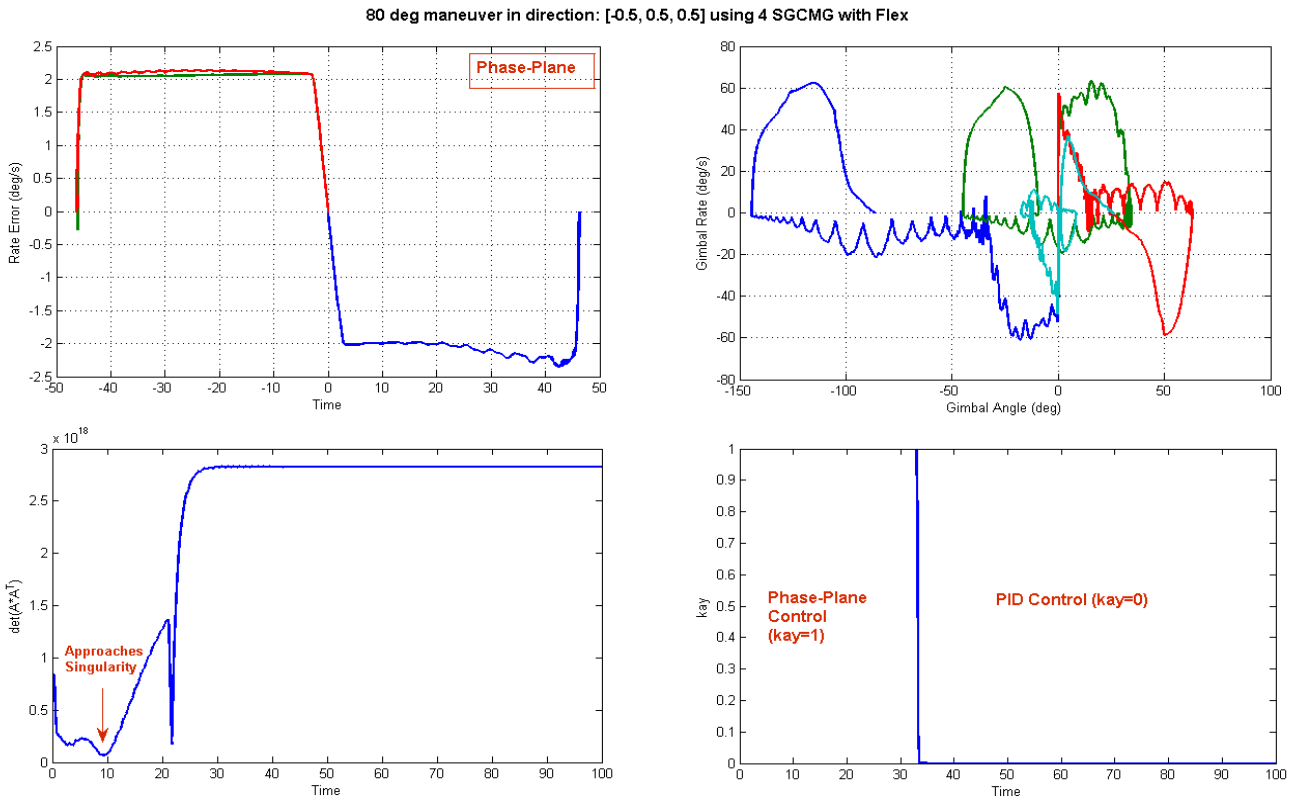


Figure 3.4.3.6(a) Spacecraft performs an 80 (deg) eigenaxis maneuver


80 deg maneuver in direction: [-0.5, 0.5, 0.5] using 4 SGCMG with Flex
CMG Momentum in Body (ft-lb-sec)
Time
CMG Control Torque (ft-lb)
Time
Roll
Pitch
Yaw
body rates (deg/s)
Time
x 10^-12
System Momentum, (ft-lb-sec)
Roll
Pitch
Yaw
Magnitude
Time


80 deg maneuver in direction: [-0.5, 0.5, 0.5] using 4 SGCMG with Flex
Phase-Plane
Rate Error (deg/s)
Time
Gimbal Rate (deg/s)
Gimbal Angle (deg)
x 10^18
det(A*A^T)
Approaches Singularity
Time
kay
Phase-Plane Control (kay=1)
PID Control (kay=0)
Time

Case # 2, a 100° Maneuver in direction: (1, 1, -1)

This is a 100 (deg) maneuver in a different direction showing similar results.

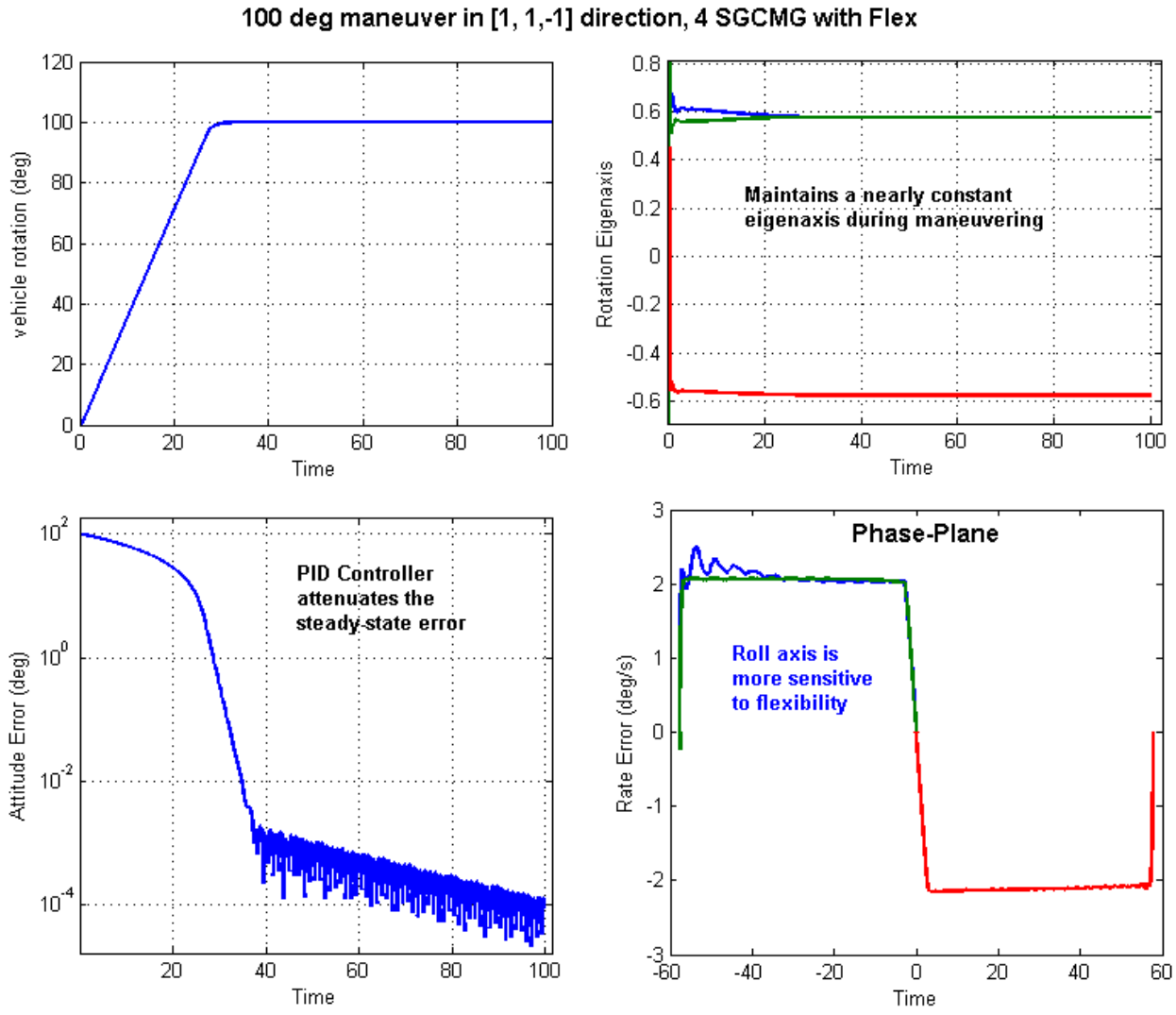


Figure 3.4.3.7(a) Spacecraft performs a 100 (deg) eigenaxis maneuver

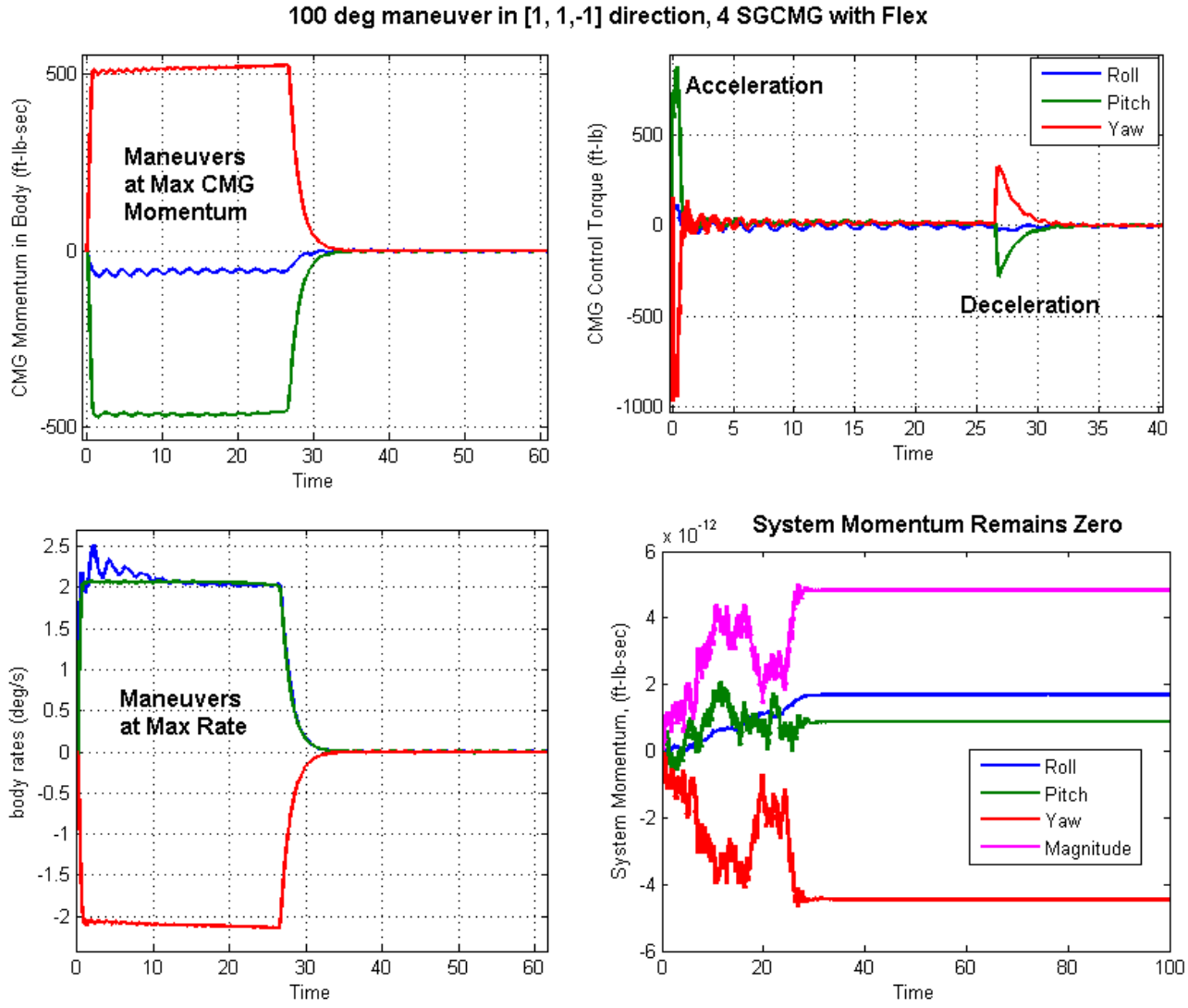


Figure 3.4.3.7(b) Spacecraft performs the maneuver at max rate and CMG momentum. System momentum remains constant (zero)

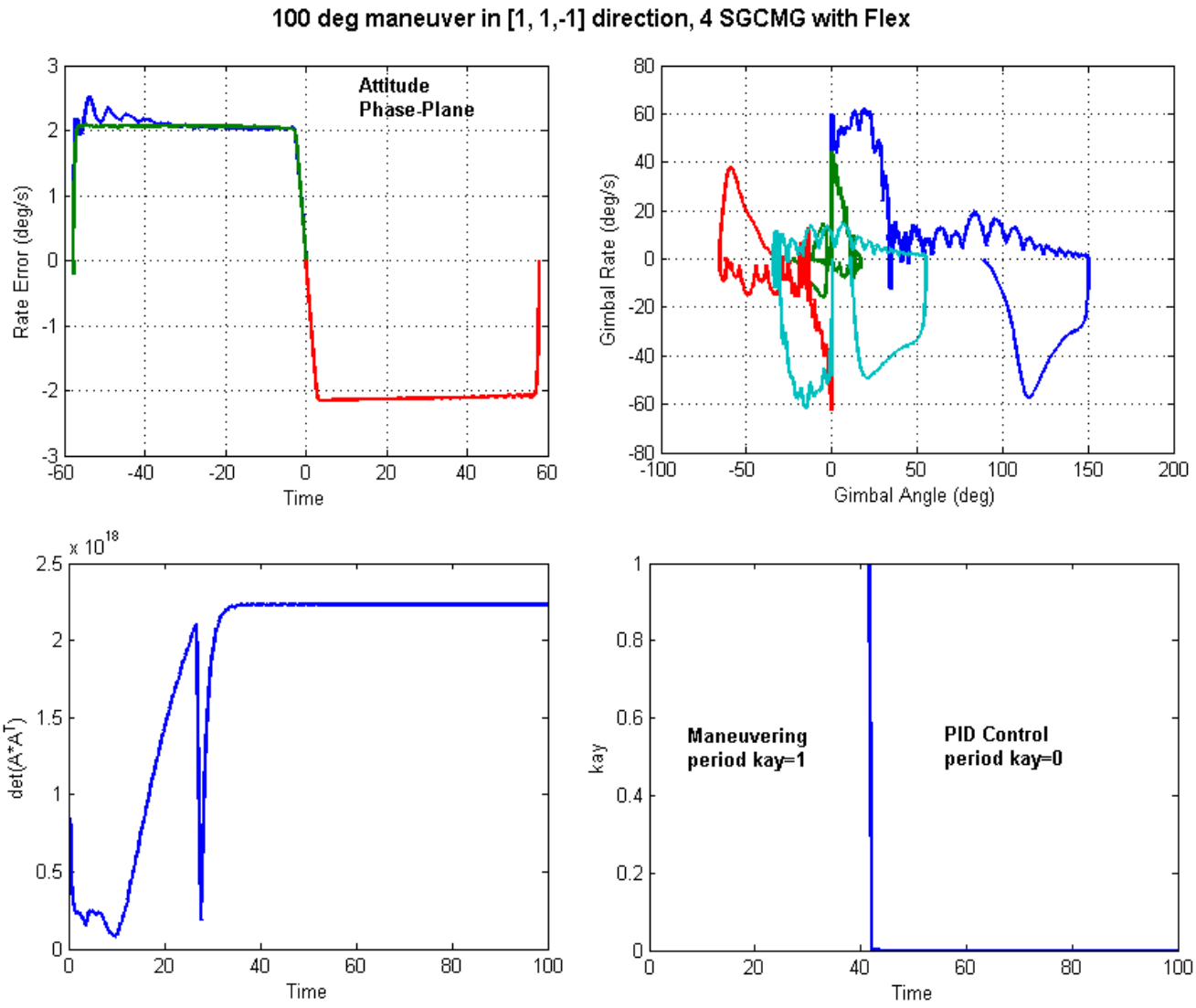


Figure 3.4.3.7(c) When the error is sufficiently small the ACS switches to PID control

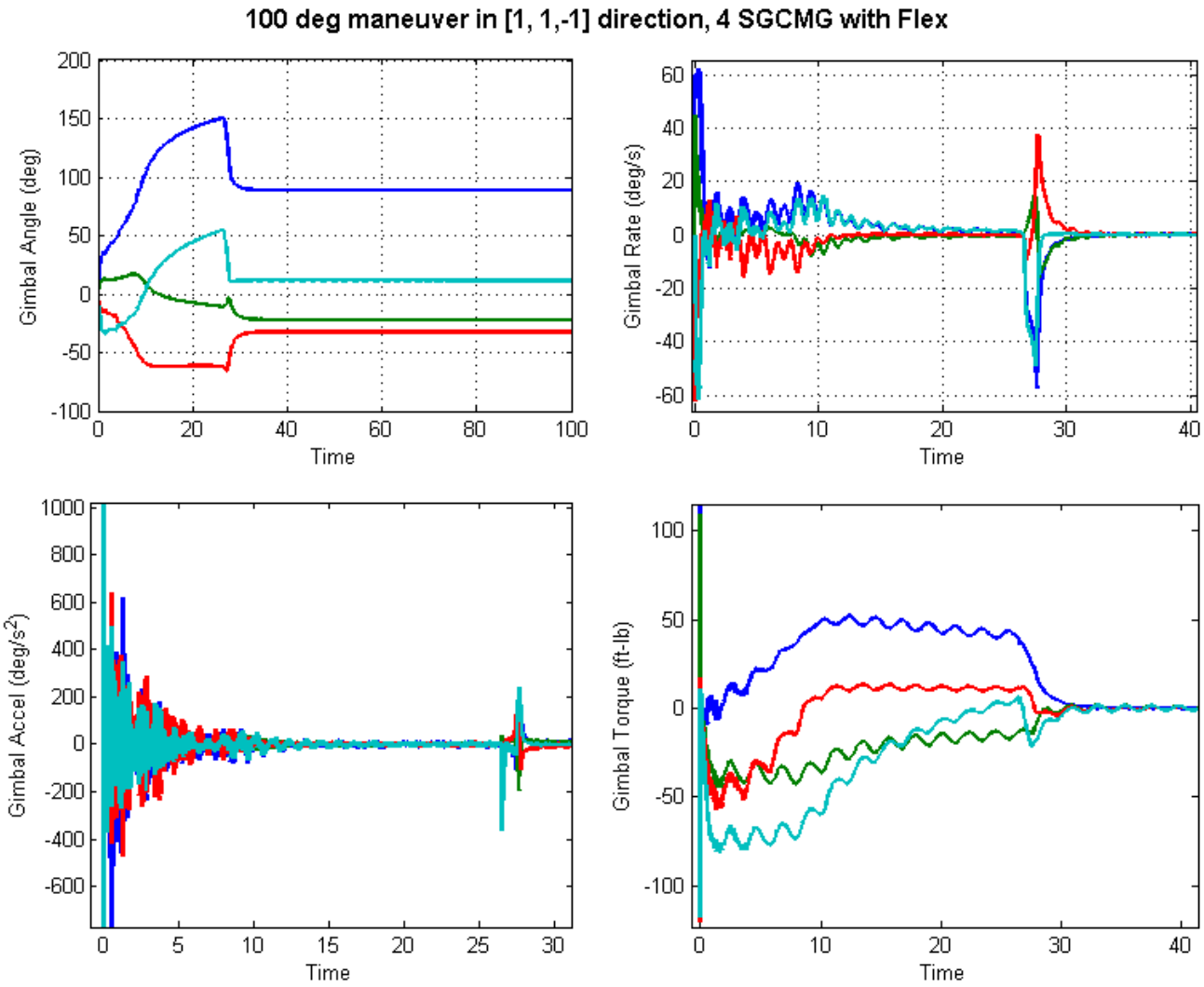


Figure 3.4.3.7(d) CMG Gimbal Activity

Case # 3, a 120° Maneuver in direction: (-0.2, 0.4, 0.4)

This maneuver hits a singularity at 10 (sec). The singularity avoidance system prevents a gimbal lock at the expense of a very small disturbance on the spacecraft. The singularity encounter, however, did not degrade the quality of the maneuver.

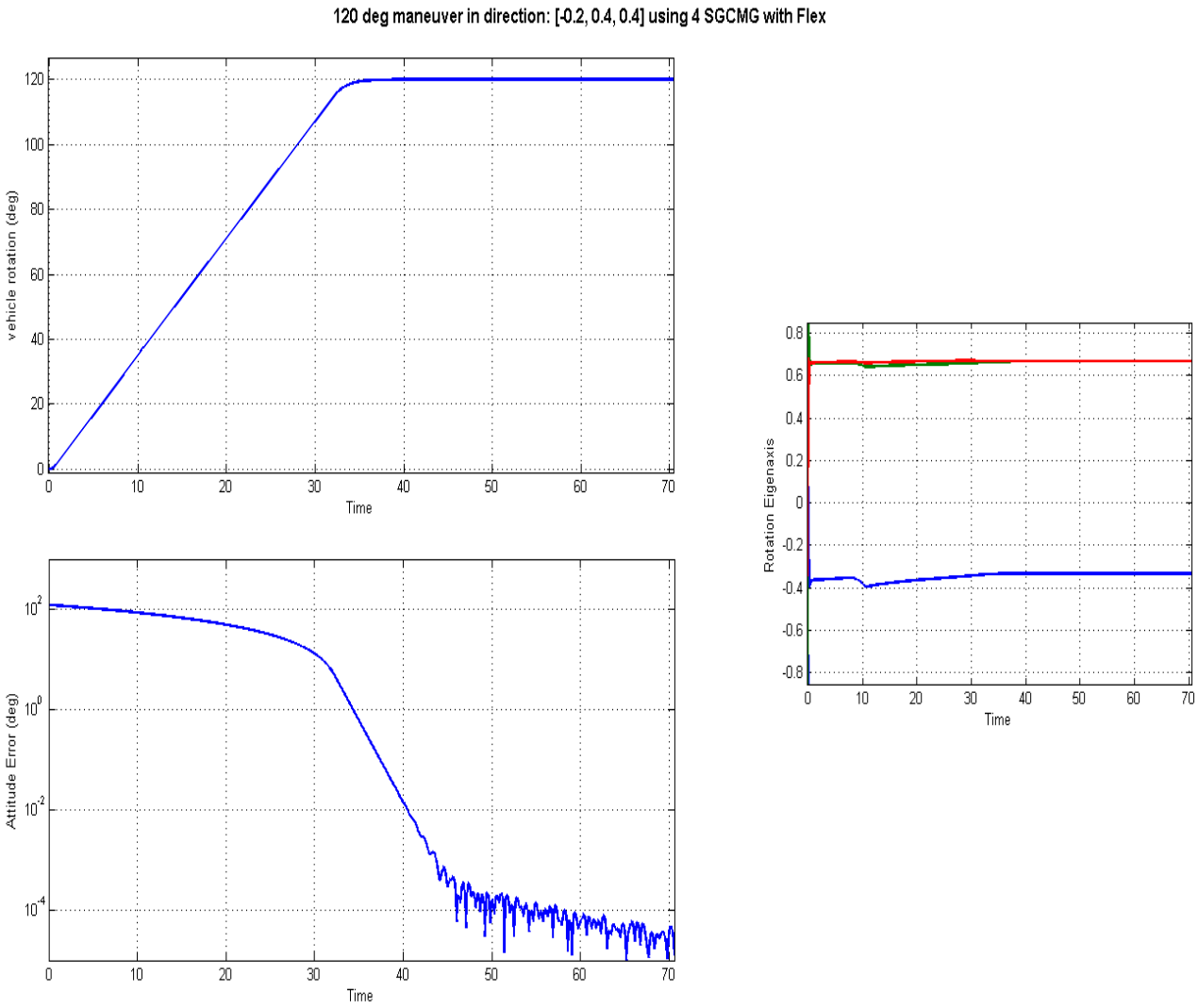
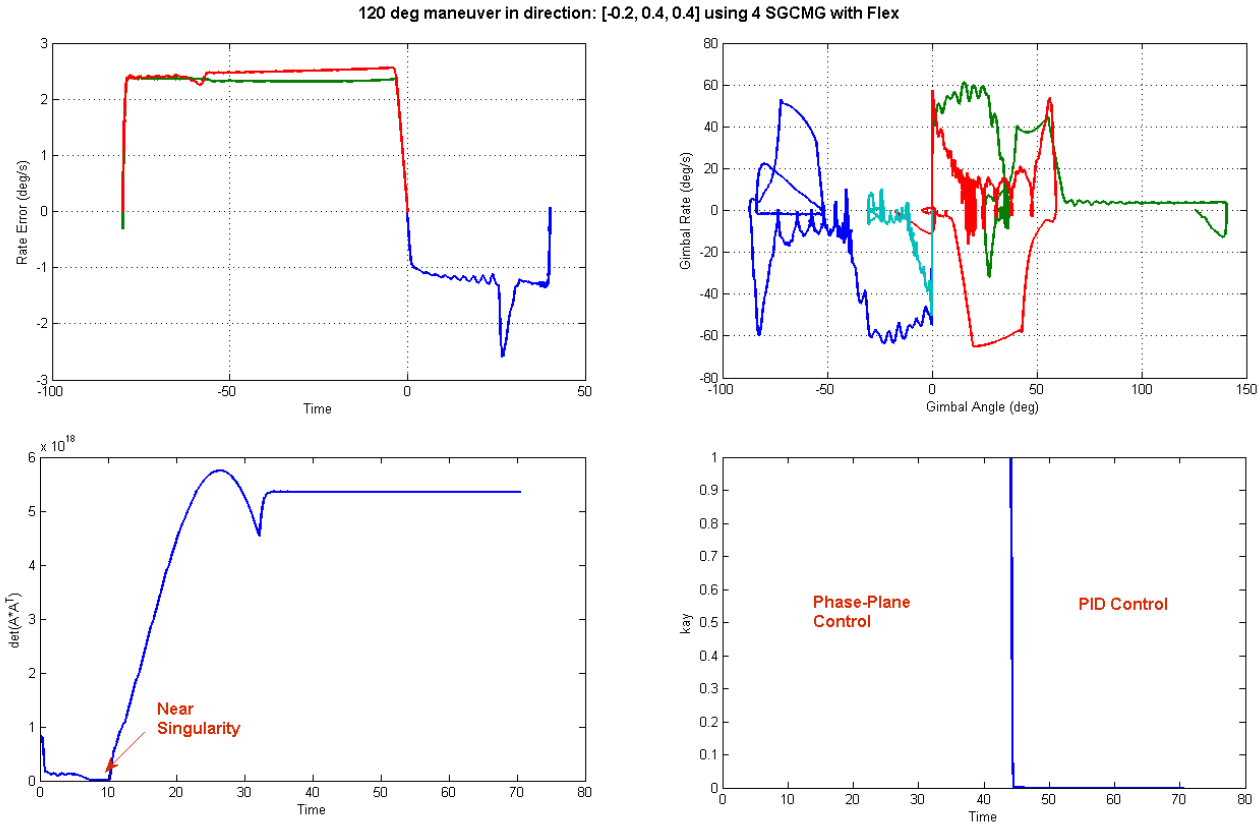
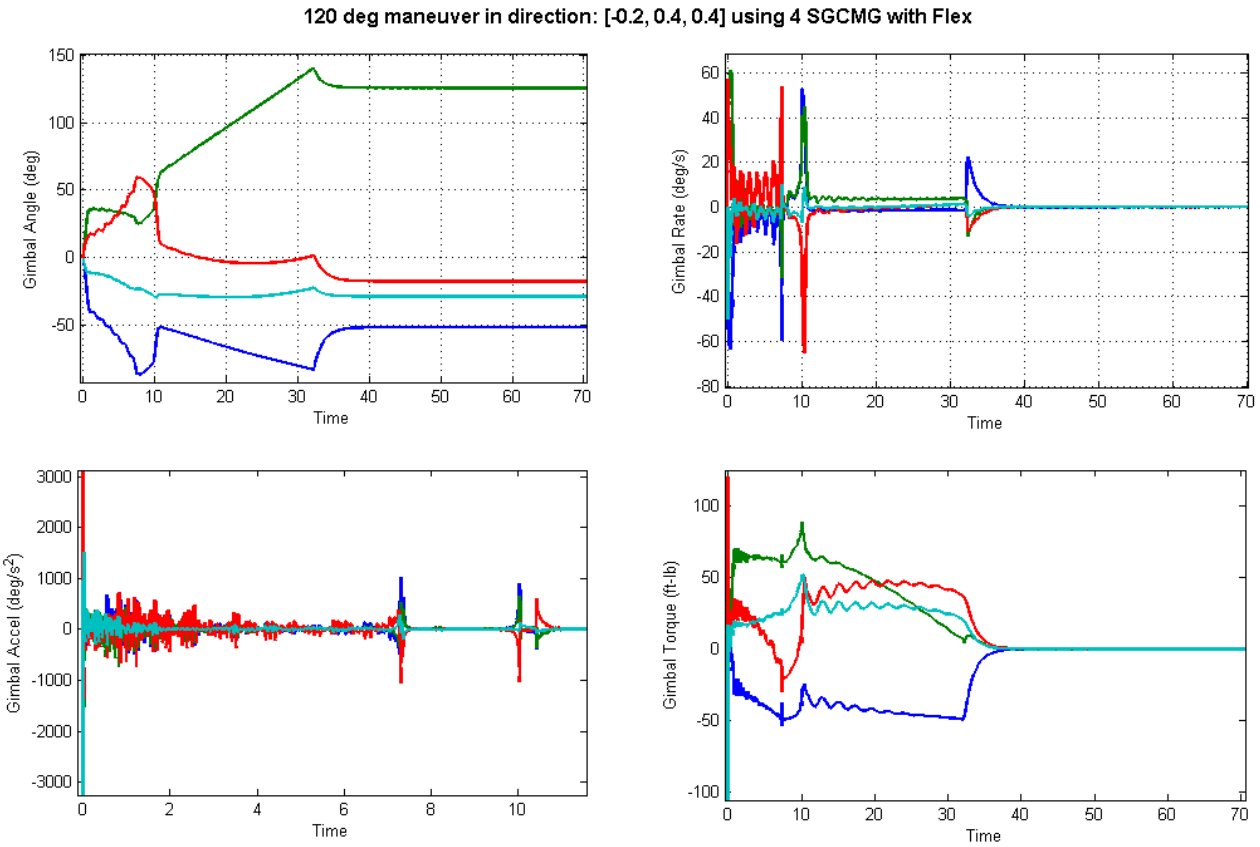


Figure 3.4.3.8(a) The singularity causes a small dent on the eigenaxis


120 deg maneuver in direction: [-0.2, 0.4, 0.4] using 4 SGCMG with Flex
Rate Error (deg/s)
Time
Gimbal Rate (deg/s)
Gimbal Angle (deg)
x 10^18
det(A*A^T)
Near
Singularity
Time
kay
Phase-Plane
Control
PID Control
Time


120 deg maneuver in direction: [-0.2, 0.4, 0.4] using 4 SGCMG with Flex
Gimbal Angle (deg)
Time
Gimbal Rate (deg/s)
Time
Gimbal Accel (deg/s^2)
Time
Gimbal Torque (ft-lb)
Time

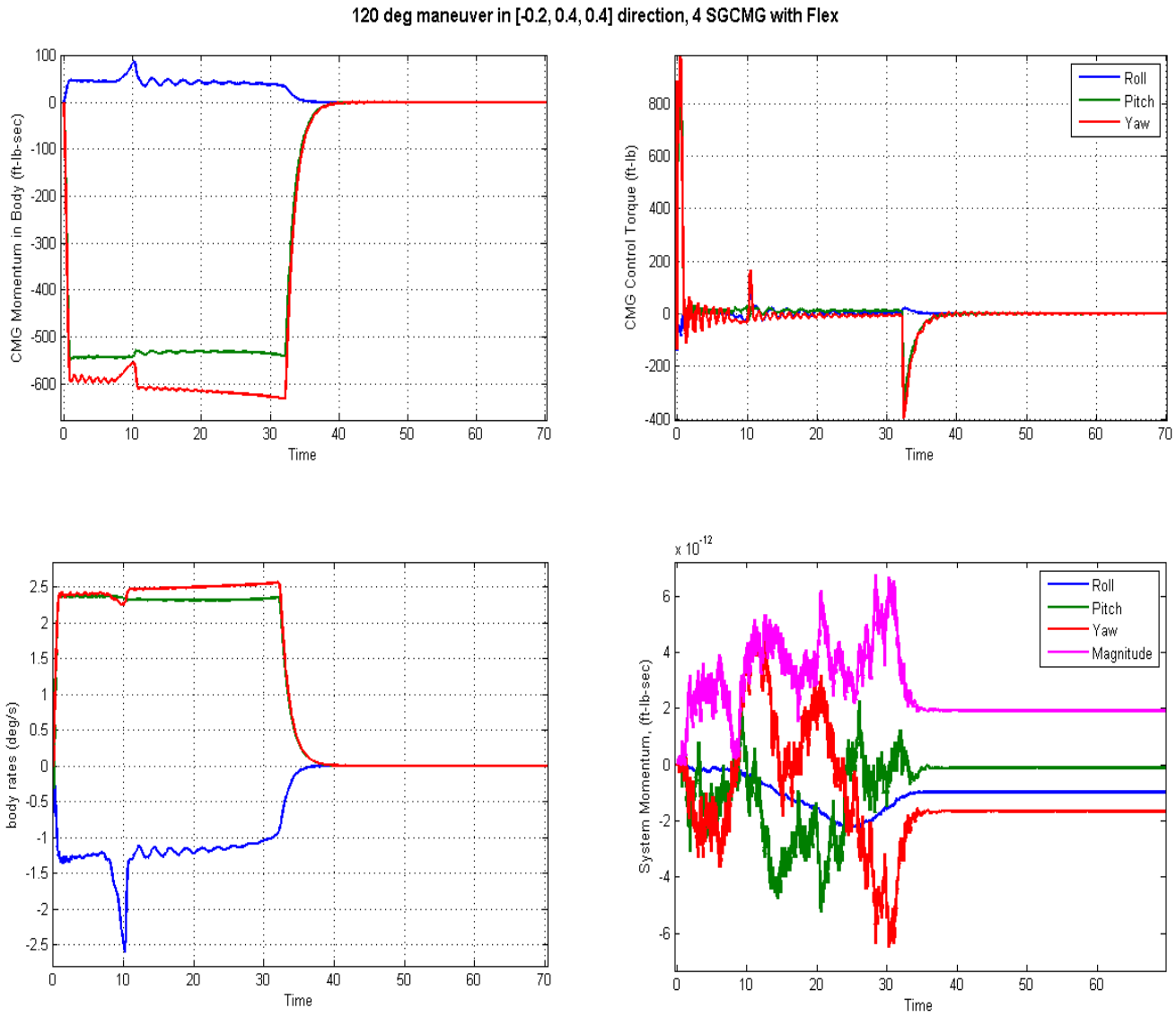


Figure 3.4.3.8(c) The singularity causes a small disturbance on the spacecraft as the singularity avoidance logic attempts to steer the vehicle around it.

Frequency Response Stability Analysis

The following model "*Open_Loop.mdl*" is similar to the non-linear simulation model but it has the control laws linearized and it is used to perform linear stability analysis. It uses the linearized functions "*Lin_ACS.m*" for attitude control, and "*Lin_Steering.m*" for steering. The control loop is broken for frequency response analysis at the rate error command to the steering logic. Only two of the control loops, pitch, or yaw, are analyzed. In the yaw analysis example shown in Figure 3.4.3.8, the yaw loop is opened, but the roll and pitch loops are closed. The Matlab file "freq.m" calculates the frequency response between the input and the output and plots the Bode and the Nichols plots. The stability analysis plots are shown in Figure 3.4.3.9. Stability is measured by the phase and gain margins from the red cross. This model is also used for tuning the PID gains to maximize stability. The ACS bandwidth was reduced from the previous rigid-body analysis to avoid exciting low frequency flex modes to instability. Compensator filters were also included in the ACS. A low-pass filter was also included to filter out the error signal which turns on the PID to provide a smoother transition between maneuvering and PID control.

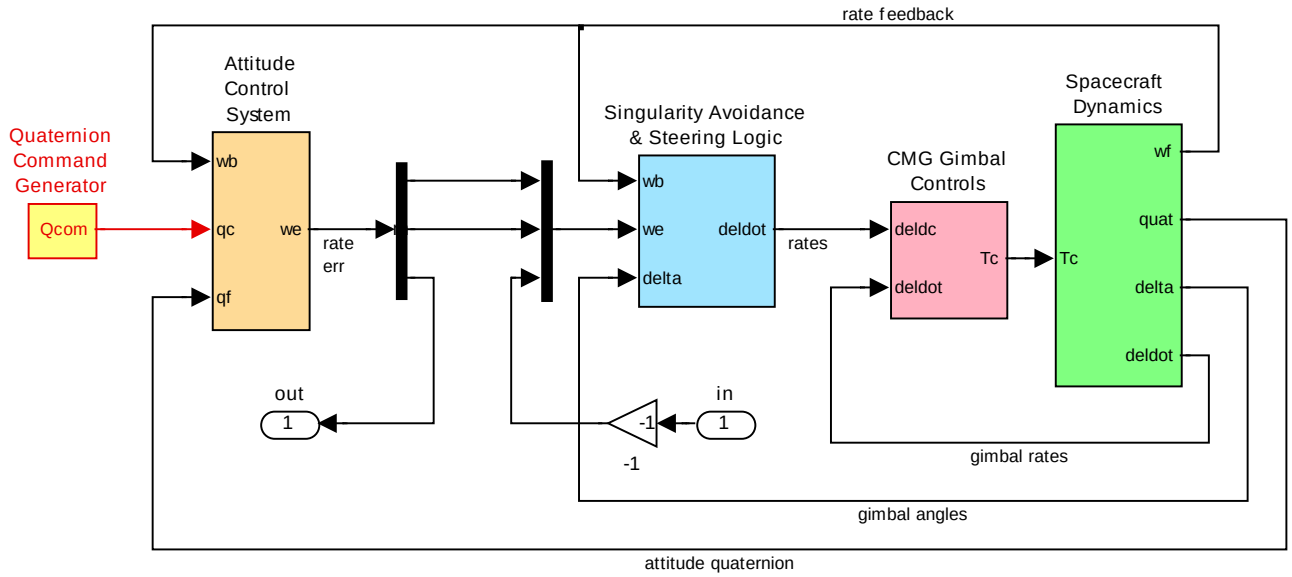


Figure 3.4.3.8 Simulink Model "Open_Loop.mdl" used for frequency response stability analysis, shown in this case for yaw axis analysis

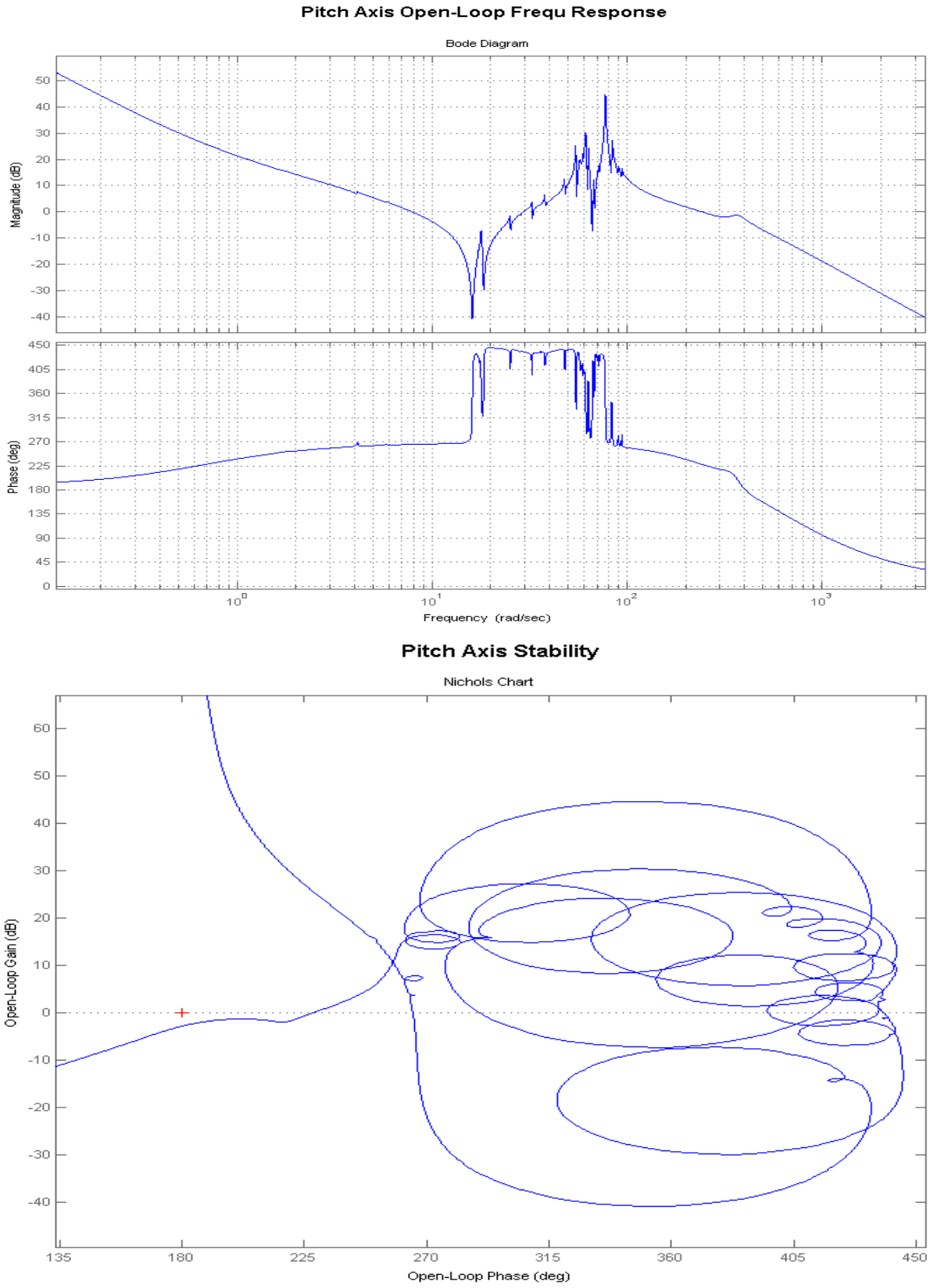


Figure 3.4.3.9a Bode and Nichols Plots showing stability margins of the Pitch axis PID system

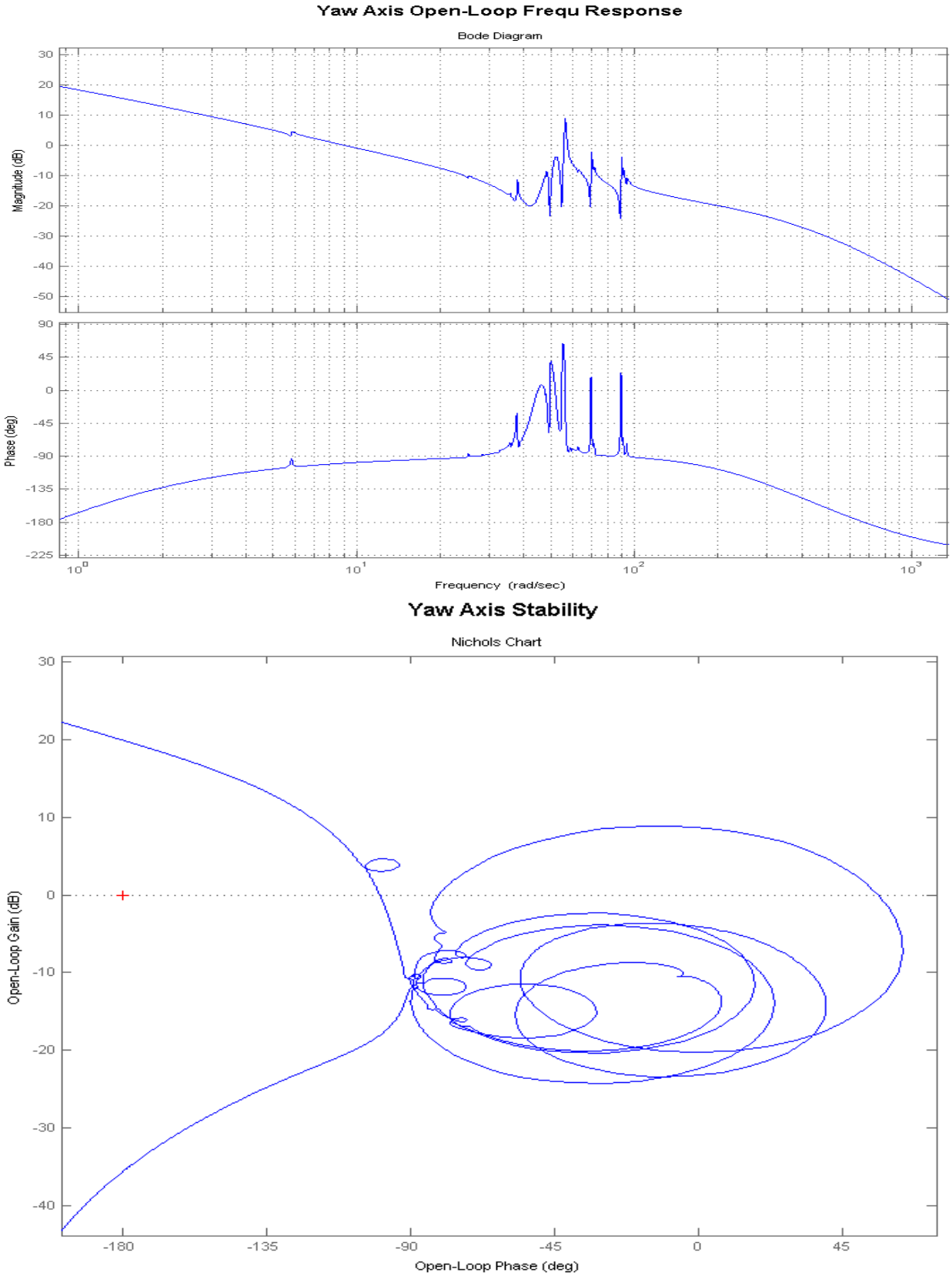


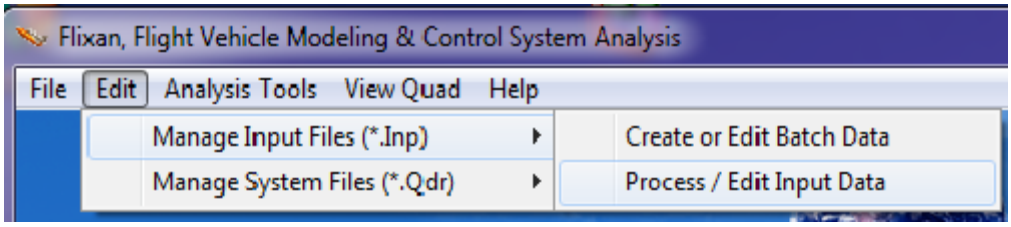
Figure 3.4.3.9b Bode and Nichols Plots showing stability margins of the Yaw axis PID system

3.5 Using the Flight Vehicle Modeling Program to Analyze the Four SGCMG System

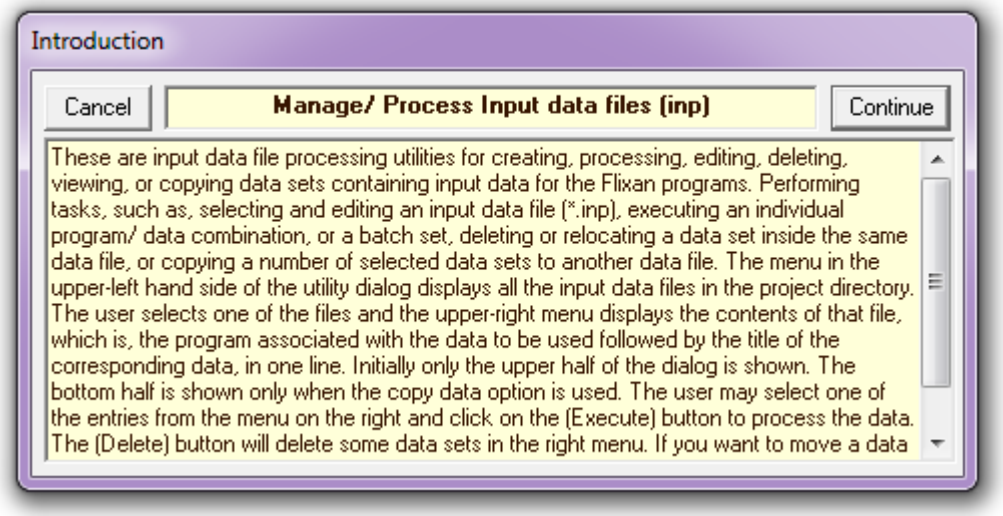
In the previous section we used spacecraft and reaction wheel non-linear models developed in Matlab and then we coupled them with state-space flexibility models that were developed using the Flixan Flex Spacecraft program. Now we will use the Flixan Flight Vehicle Modeling Program (FVP) to create similar models of the spacecraft coupled with four single gimbal CMGs, including structural flexibility, and hopefully the results will match with the results obtained from the previous section in folder "*(c) Flex 4SGCMG ACS w Gimbal*". We will first show how to obtain the spacecraft state-space models by using the existing data files and running the FVP in batch mode, and then go into details to show how we create the spacecraft and flex data from scratch. The data for this analysis is in directory "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(e) 4SGCMG using FVP* ". The input data file is "*FlexSc_CMG_FVP.Inp*" the systems file is "*FlexSc_CMG_FVP.Qdr*". The modal data files are the same as previously in the RCS analysis "*FlexSc_FEM.Mod*" and "*FlexSc_FEM.Nod*". The input data file contains data for two spacecraft with 4 SGCMGs, a rigid model and a flex model. A set of modal data consisting of 40 selected flex modes is included at the bottom of the input data file. Some unused states and outputs are eliminated using the Flixan truncation utility. The Matlab analysis is performed in the sub-directory "Matan".

Creating the Systems in Batch Mode

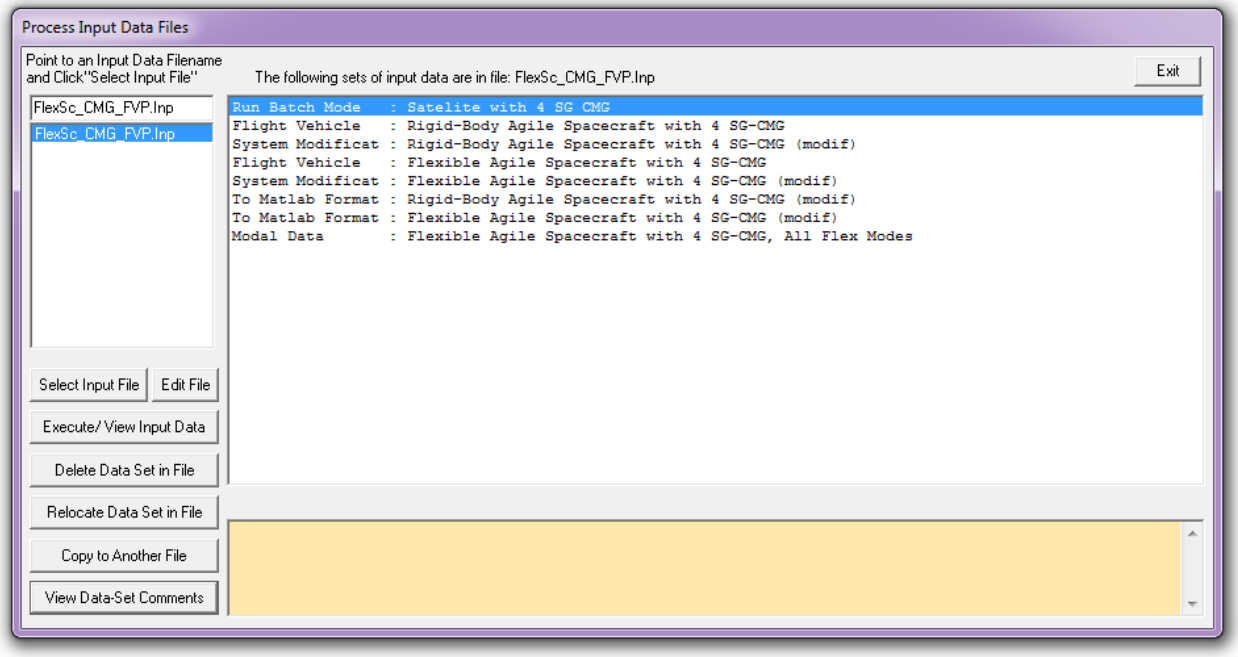
On the top of the input data file there is a batch data-set "*Batch for Spacecraft with 4 SG CMG*". It is a short script of commands that speeds up the generation of the spacecraft systems. To run the batch, first start the Flixan program, go to folder "*…Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(e) 4SGCMG using FVP*". Go to "*Edit*", "*Manage Input Files*", and then "*Process/ Edit Input Data*".



When the following dialog appears, click "Continue".



From the following dialog select the input file "*FlexSc_CMG_FVP.Inp*" and press the "*Select Input File*" button. The menu on the right shows all the data-sets that are saved inside the input file. Select the top batch set: "*Batch for Spacecraft with 4 SG CMG*", and click on "*Execute/ View Input Data*".

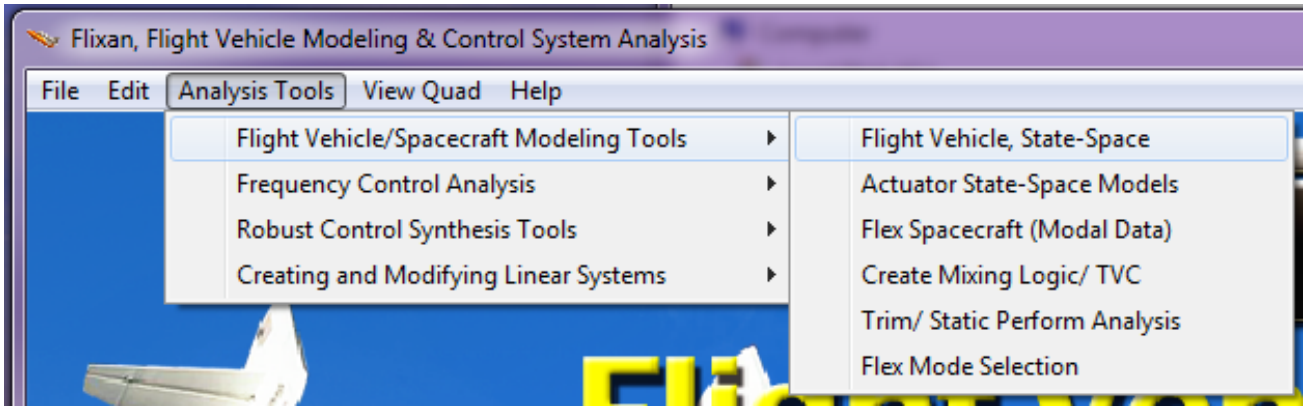


The Flixan program executes all the data-sets which are called by the batch and saves the spacecraft systems in file "*FlexSc_CMG_FVP.Qdr*". It also creates two Matlab state-space m-files for the spacecraft with four CMGs that can be loaded into Matlab: a rigid-body model in file "*sc_4cmg_rb.m*" and a flex spacecraft model "*sc_4cmg_flex.m*". Click "Exit" to end the Flixan program. The two files are transferred to subfolder "Matan" for analysis.

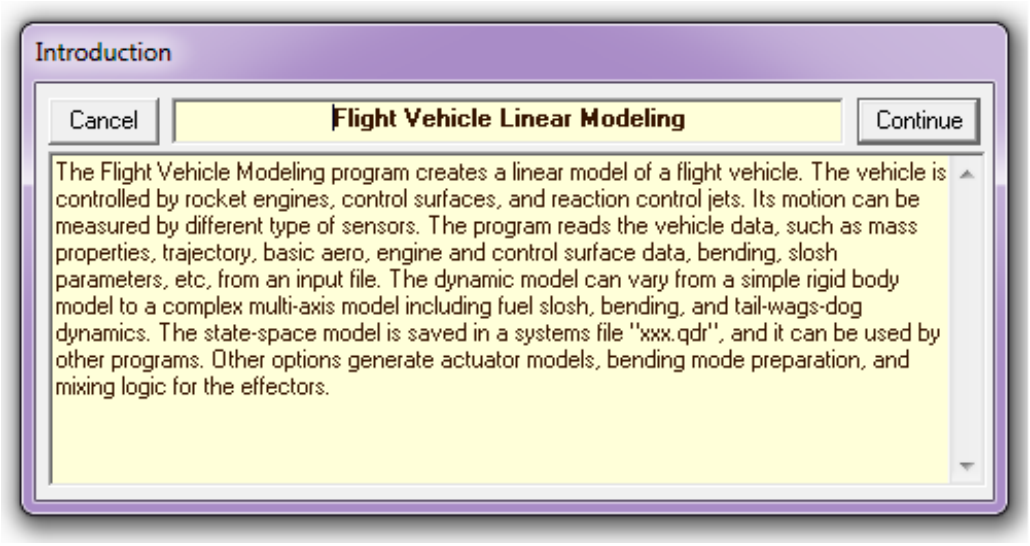
Creating the Spacecraft Input Data from Scratch

But what if we don't already have the input data in file "*FlexSc_CMG_FVP.Inp*"? What if we want to create a new set of spacecraft data from scratch in an empty "*FlexSc_CMG_FVP2.Inp*" file and create the state-space model directly from the new file? I personally don't like to use the Flixan utility menus to create new systems not because they are not efficient but because I make mistakes or forget something and I have to do it all over again. I usually copy old models and modify them using standard editors.

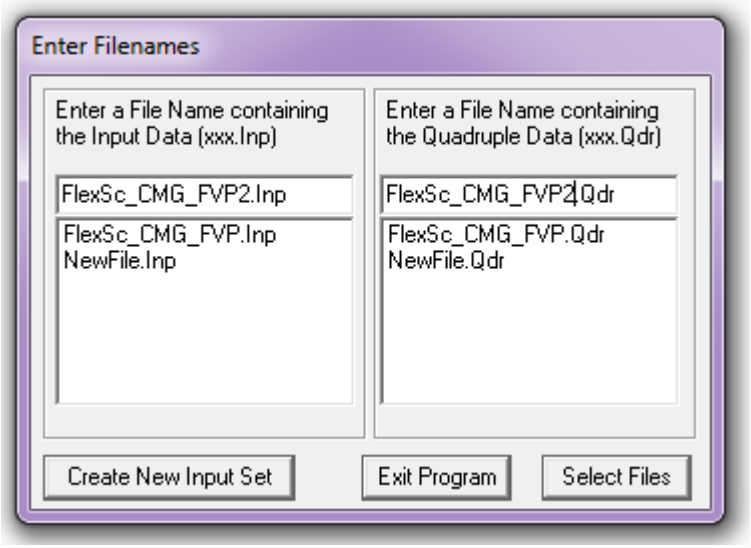
To create the spacecraft data from scratch, first start the Flixan program and select the directory directory "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(e) 4SGCMG using FVP* ". Then go to "Analysis Tools", "Flight Vehicle/ Spacecraft Modeling Tools", and "Flight Vehicle State-Space", as shown below.

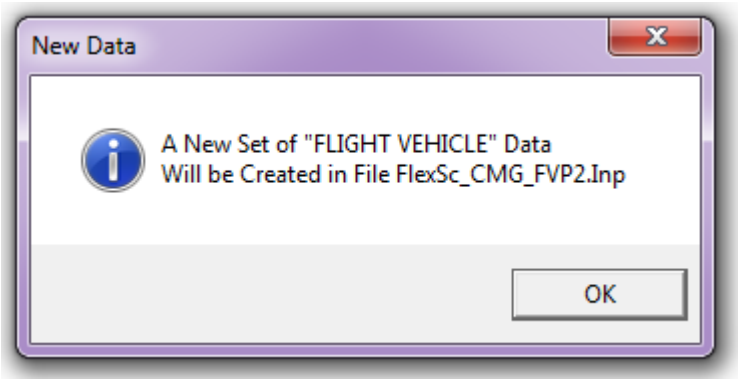


In the following introductory menu, press "Continue".

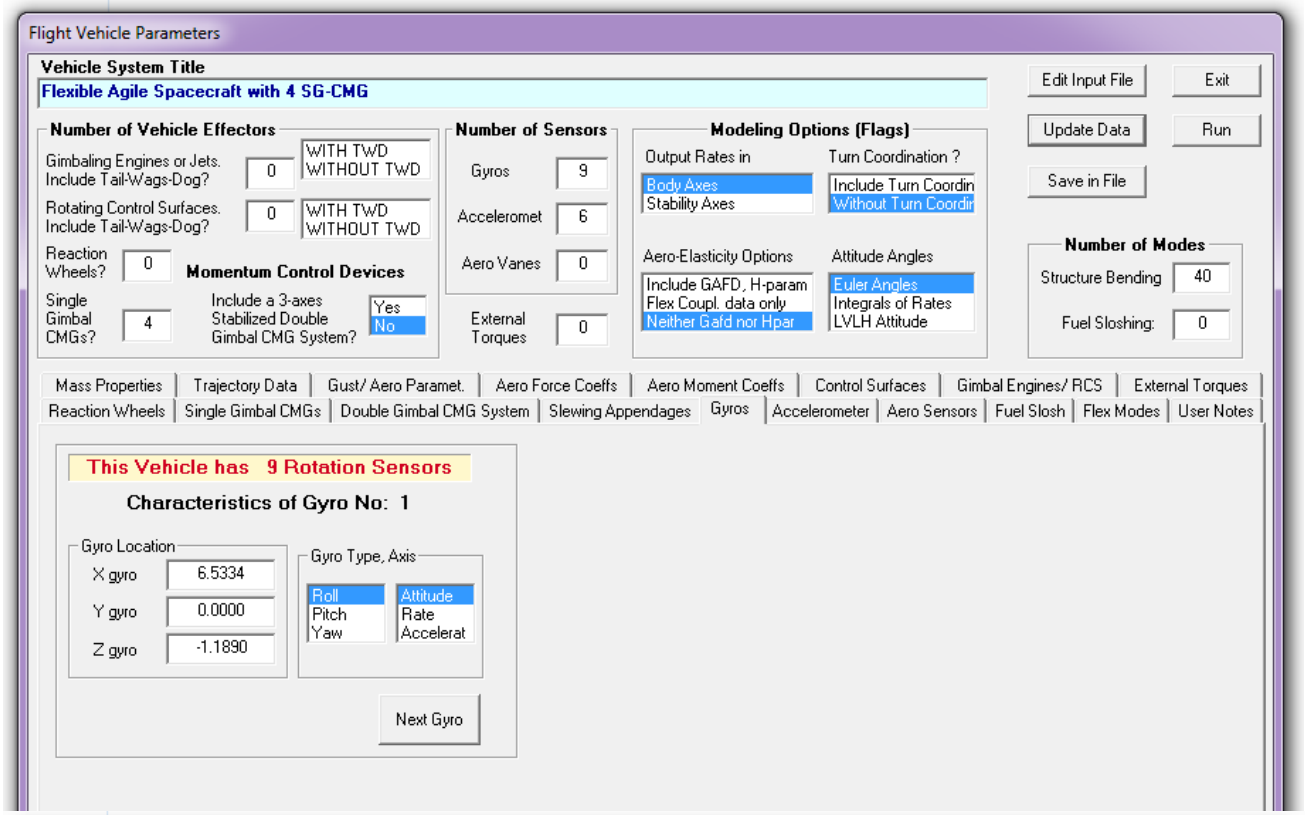


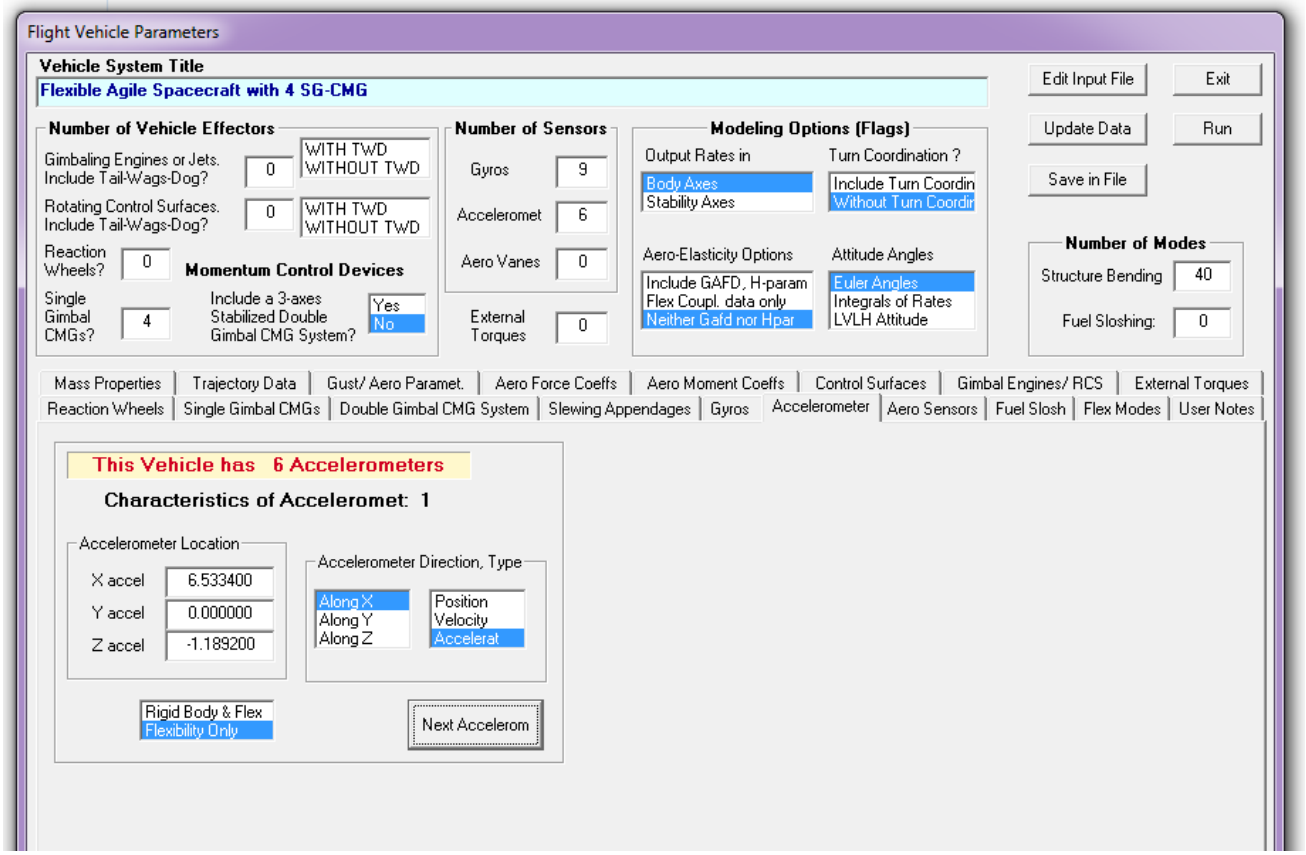
The following is a file selection menu for the input and output data files. Since in this case we assume that we don't have any input data, enter a new input file name "*FlexSc_CMG-FVP2.Inp*", and a new output file name "*FlexSc_CMG-FVP2.Qdr*", that will be created and click on "*Create a New Input Set*".



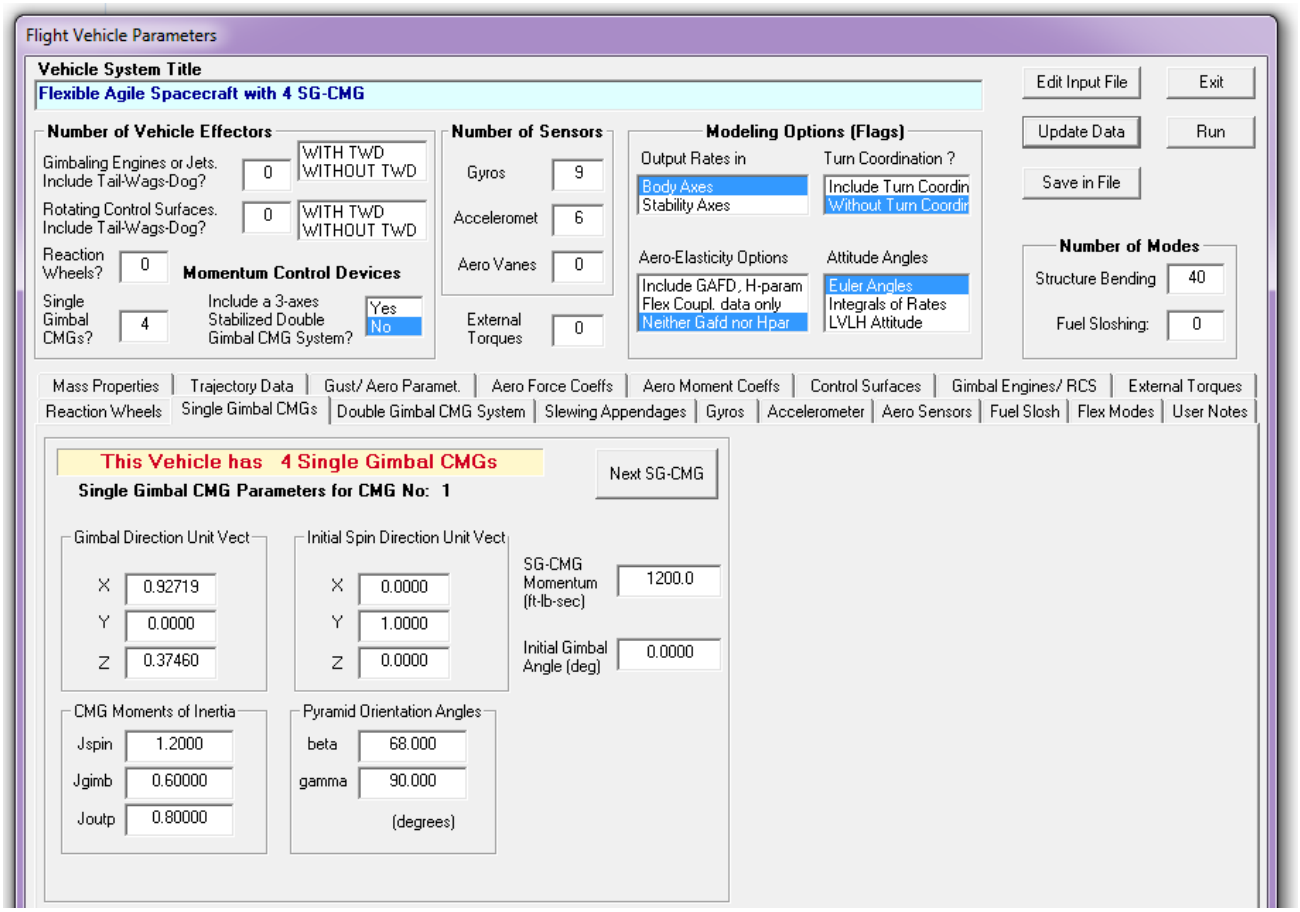


The Flixan program acknowledges that a new set of data will be created and it will activate a data entering utility where the user can enter the spacecraft data interactively by means of menus and dialogs, as shown below. Using the following flight vehicle data entering dialog first enter the title on the top, define the number of SG-CMGs, the number of gyros and accelerometers, set some of the menus as shown, enter also 40 flex modes and click "Update Data" to keep the data in memory. Then go to some of the tabs at the bottom. Click on the gyros tab and start entering the gyro data. For gyro #1 enter its location in vehicle coordinates, the direction of measurement "roll", and specify that it is an attitude and not a rate measurement. Click on "Next Gyro" to enter the data for gyro #2, and so on for the 9 gyros specified at the top. Remember to click on "Update Data" before moving to the next tab. Now select another tab. Let's say the accelerometers, and start entering the data for the first accelerometer which is measuring acceleration in the x direction. Enter also its location in spacecraft coordinates. Click on "Next Accelerometer" to enter the data for accelerometer #2, and so on for all six accelerometers. Remember to click on "Update Data" before moving to the next tab. Click on the "Mass Properties" tab and do the same.


Flight Vehicle Parameters
Vehicle System Title
Flexible Agile Spacecraft with 4 SG-CMG
Edit Input File
Exit
Update Data
Run
Save in File
Number of Vehicle Effectors
Gimbaling Engines or Jets. Include Tail-Wags-Dog?
0
WITH TWD
WITHOUT TWD
Rotating Control Surfaces. Include Tail-Wags-Dog?
0
WITH TWD
WITHOUT TWD
Reaction Wheels?
0
Momentum Control Devices
Single Gimbal CMGs?
4
Include a 3-axes Stabilized Double Gimbal CMG System?
Yes
No
Number of Sensors
Gyros
9
Accelerometer
6
Aero Vanes
0
External Torques
0
Modeling Options (Flags)
Output Rates in
Body Axes
Stability Axes
Turn Coordination ?
Include Turn Coordin
Without Turn Coordir
Aero-Elasticity Options
Include GAFD, H-param
Flex Coupl. data only
Neither Gafd nor Hpar
Attitude Angles
Euler Angles
Integrals of Rates
LVLH Attitude
Number of Modes
Structure Bending
40
Fuel Sloshing:
0
Mass Properties
Trajectory Data
Gust/ Aero Paramet.
Aero Force Coeffs
Aero Moment Coeffs
Control Surfaces
Gimbal Engines/ RCS
External Torques
Reaction Wheels
Single Gimbal CMGs
Double Gimbal CMG System
Slewing Appendages
Gyros
Accelerometer
Aero Sensors
Fuel Slosh
Flex Modes
User Notes
This Vehicle has 9 Rotation Sensors
Characteristics of Gyro No: 1
Gyro Location
X gyro
6.5334
Y gyro
0.0000
Z gyro
-1.1890
Gyro Type, Axis
Roll
Pitch
Yaw
Attitude
Rate
Accelerat
Next Gyro


Flight Vehicle Parameters
Vehicle System Title
Flexible Agile Spacecraft with 4 SG-CMG
Edit Input File
Exit
Update Data
Run
Save in File
Number of Vehicle Effectors
Gimbaling Engines or Jets. Include Tail-Wags-Dog?
0
WITH TWD
WITHOUT TWD
Rotating Control Surfaces. Include Tail-Wags-Dog?
0
WITH TWD
WITHOUT TWD
Reaction Wheels?
0
Momentum Control Devices
Single Gimbal CMGs?
4
Include a 3-axes Stabilized Double Gimbal CMG System?
Yes
No
Number of Sensors
Gyros
9
Accelerometer
6
Aero Vanes
0
External Torques
0
Modeling Options (Flags)
Output Rates in
Body Axes
Stability Axes
Turn Coordination ?
Include Turn Coordin
Without Turn Coordir
Aero-Elasticity Options
Include GAFD, H-param
Flex Coupl. data only
Neither Gafd nor Hpar
Attitude Angles
Euler Angles
Integrals of Rates
LVLH Attitude
Number of Modes
Structure Bending
40
Fuel Sloshing:
0
Mass Properties
Trajectory Data
Gust/ Aero Paramet.
Aero Force Coeffs
Aero Moment Coeffs
Control Surfaces
Gimbal Engines/ RCS
External Torques
Reaction Wheels
Single Gimbal CMGs
Double Gimbal CMG System
Slewing Appendages
Gyros
Accelerometer
Aero Sensors
Fuel Slosh
Flex Modes
User Notes
This Vehicle has 6 Accelerometers
Characteristics of Accelerometer: 1
Accelerometer Location
X accel
6.533400
Y accel
0.000000
Z accel
-1.189200
Accelerometer Direction, Type
Along X
Along Y
Along Z
Position
Velocity
Accelerat
Rigid Body & Flex
Flexibility Only
Next Accelerom

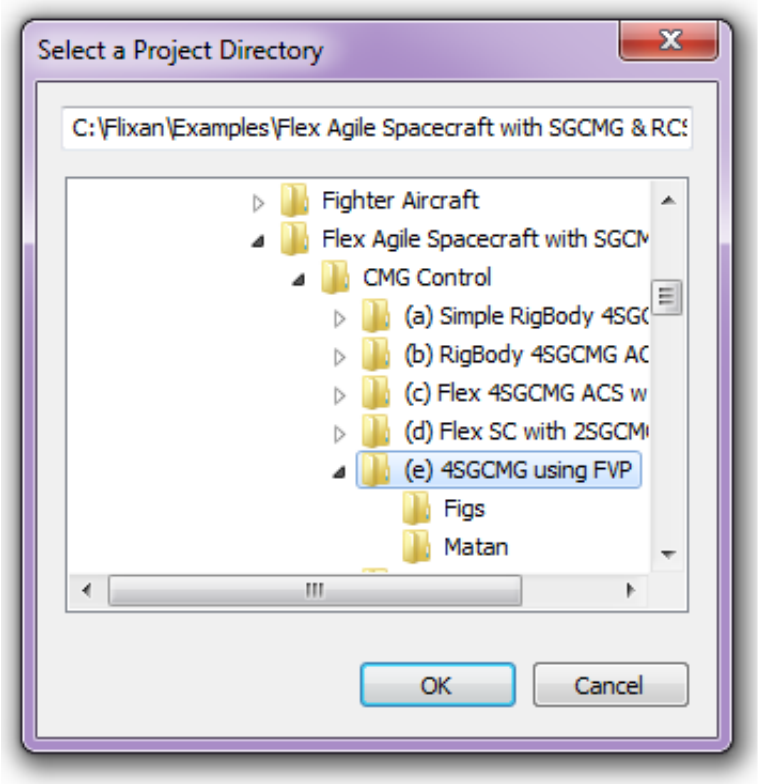
Now click on the "Single Gimbal CMG" tab to open it and enter the data required for the four SG-CMGs. That is, starting from the first one: the gimbal direction vector, the momentum reference direction (which in this case it is the same as the initial momentum direction), the CMG moments of inertia about spin, gimbal, and output axes, the CMG orientation angles (β and γ) in the pyramid as already defined, the constant CMG momentum magnitude, and the initial gimbal angle (δ_0). Click on "Next-SG-CMG" to enter the data for the second CMG, and so on. Remember to click on "Update Data" when you finish with the SG-CMGs. You may also click on the "*User Notes*" tab to enter some notes that will be included as comments in both: the input data, and in the system (a,b,c,d) data, below the title.

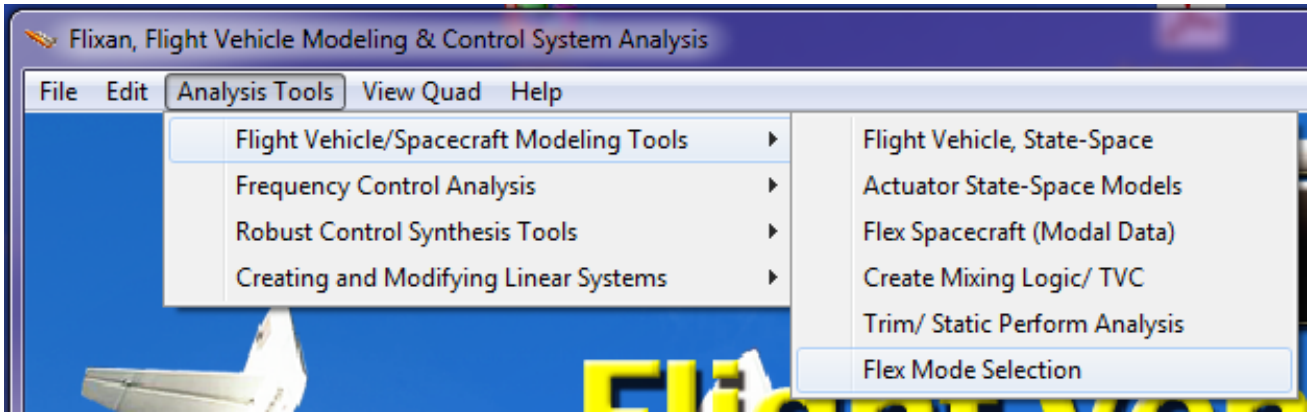


When you finish entering the spacecraft data, click on "*Save in File*" button to save the data in file "*FlexSc_CMG-FVP2.Inp*". This will create the spacecraft data set under the title "*Flexible Agile Spacecraft with 4 SG-CMG*" that you already entered. To create the spacecraft state-space model you must click on the "*Run*" button, and the program would create the spacecraft system in file "*FlexSc_CMG-FVP2.Inp*". But don't do it yet, you are not ready. You have declared that you will be using 40 structural modes but the modes have not yet been selected and saved. A set of flex modes should also be included in the input data file. It requires a lengthy mode selection process that will be discussed next. After the modes are selected and saved you can go back and rerun the Flight Vehicle Modeling program, select the same input and output files, select the title of the input data, get to the above dialog which displays the spacecraft data, and now you may click "*Run*" to create the spacecraft model that will be saved in file "*FlexSc_CMG-FVP2.Qdr*".

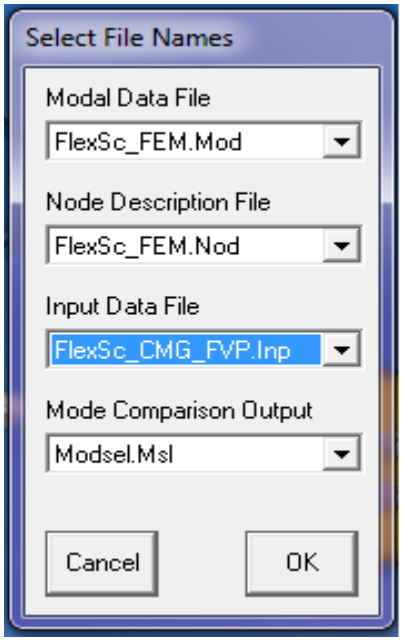
Creating Set of Selected Modal Data

The flex model requires a set of selected modes reformatted and compatible to be processed with the spacecraft data. From the original modal data in file "*FlexSc_FEM.Mod*" we must select a set of modes, scale them to match the units of the spacecraft data, and save the selected set in file "*FlexSc_CMG_FVP.Inp* ". We must first start the Flixan program and go to folder "*…Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(e) 4SGCMG using FVP*". Then go to "*Analysis Tools*", "*Flight Vehicle/ Spacecraft Modeling Tools*", and then to "*Flex Mode Selection*", as shown below.

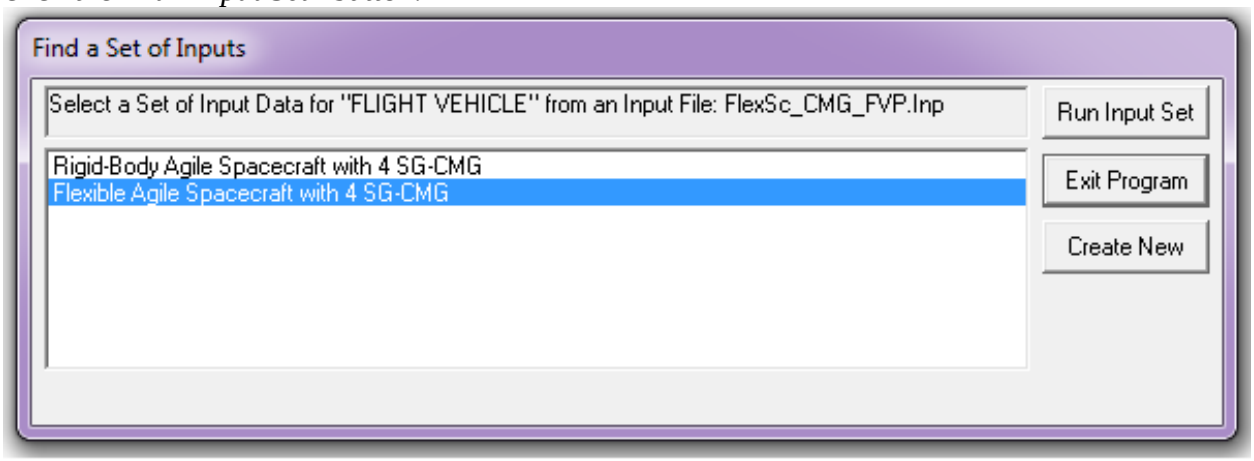




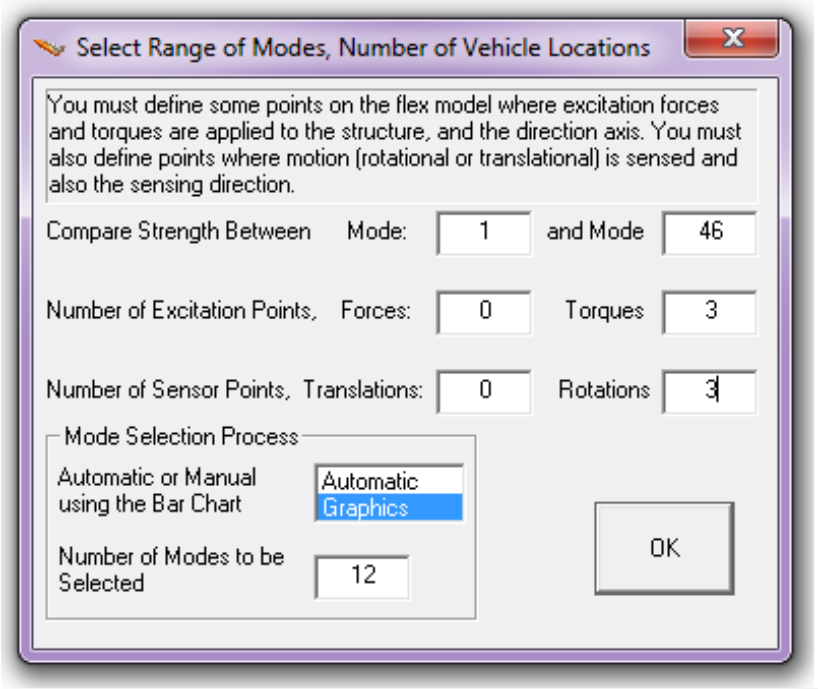
Using the filenames menu on the right we select the modal data filename which has an extension (.Mod), the nodes file (.Nod), the flight vehicle input data file (.Inp), and an output filename (Modsel.Msl), where the program will save the relative mode strength at the completion of mode selection. The modal data and nodes map files were used earlier in the RCS analysis. The input data file "*FlexSc_CMG_FVP.Inp*" is needed because the mode selection program needs the spacecraft data to match actuator and sensor locations with node points from the finite elements model. The nodes map is also used to facilitate this purpose.

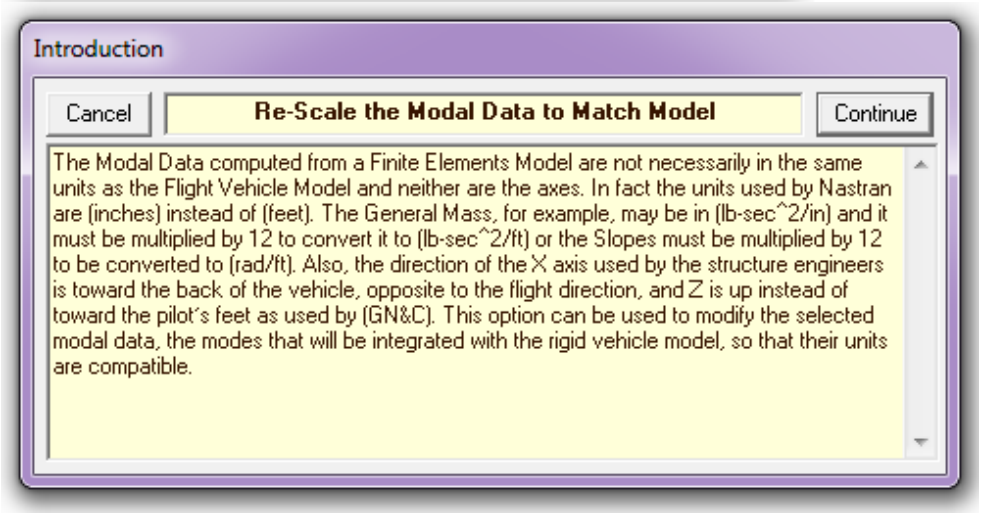


Using the menu below select the spacecraft title "*Flexible Agile Spacecraft with 4 SG-CMG*" and click the "*Run Input Set*" button.

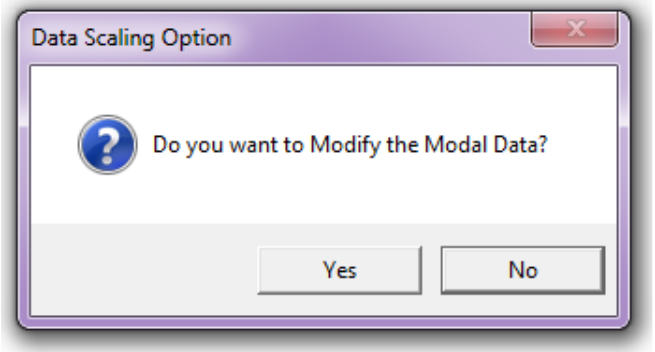


The dialog below is used to define the number of excitation, and the number of sensor points to be used for mode selection. These numbers do not have to be equal to the number of actual vehicle effectors and sensors used in the spacecraft model. It is only used for mode selection purposes. We must also define the range of modes to be evaluated (1 to 46 modes in this case). We will ignore the first 6 modes because they are rigid body modes and we should only include structural modes in the flex mode set. We do not define any forces excitations, but we define 3 torque excitations. Similarly, we do not select any translational sensors but 3 rotational sensors. These locations are only for mode selection. We will also select the graphic mode selection option where the user selects the modes from a bar chart using the mouse. The number of modes to be selected does not apply in this case. We click "OK" to continue.

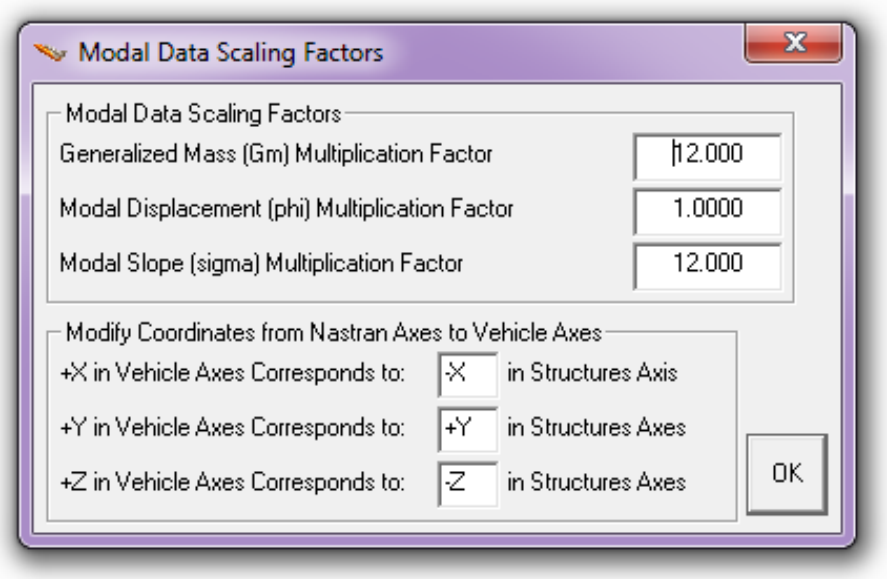




The following dialog allows you to convert the units of the modal data or to switch the direction of the coordinate axes of the finite elements model to match the rigid-body model. We click "Yes" because the modal data are in typical Nastran units and the x and z directions are reversed.



The following dialog defines the scaling factors that will be used to modify the modal shapes, slopes, generalized mass, and it will also reverse some of the directions.



The next step is to identify locations from the finite elements model (nodes) for the 3 torque excitation points and the 3 rotational sensors points that will be used in mode selection. The nodes mapping file "*FlexSc_FEM.Nod*" will be used as a reference in the process. It is included in node selection menu that helps the user to pick excitation and sensor nodes from the modal data file for mode strength comparison. For the 3 torque excitation points we select node #6, which is the CMG location, to apply torques in roll, pitch, and yaw. For the 3 rotational sensor points we select node #2, which is the location of the spacecraft gyros, to measure roll, pitch, and yaw rotations.

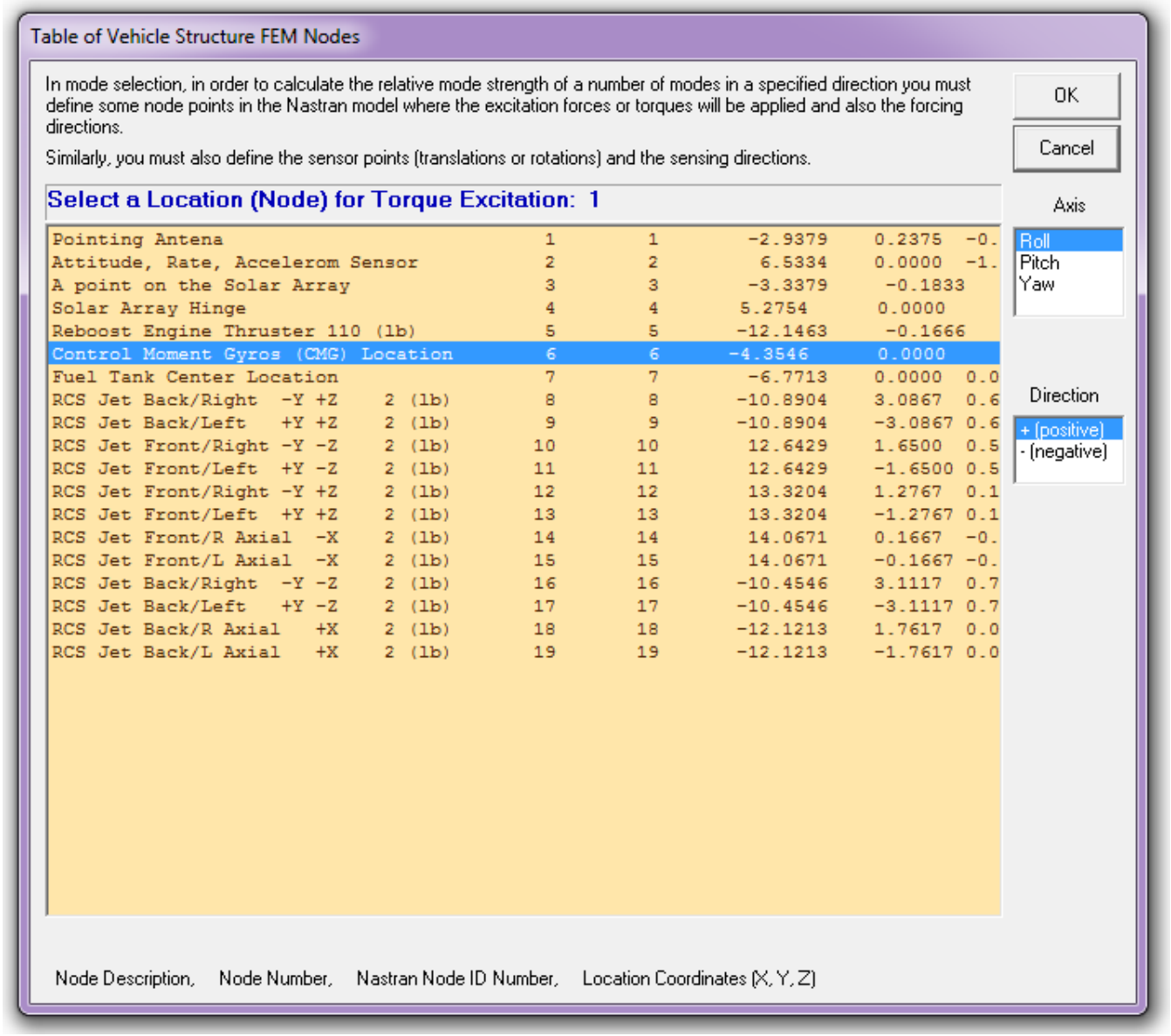


Table of Vehicle Structure FEM Nodes
In mode selection, in order to calculate the relative mode strength of a number of modes in a specified direction you must define some node points in the Nastran model where the excitation forces or torques will be applied and also the forcing directions.
Similarly, you must also define the sensor points (translations or rotations) and the sensing directions.
OK
Cancel
Select a Location (Node) for Torque Excitation: 1
Axis
Roll
Pitch
Yaw
Direction
+ (positive)
- (negative)
Pointing Antena 1 1 -2.9379 0.2375 -0.
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.0
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.6
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.6
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.5
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.5
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.1
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.1
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.7
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.7
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.0
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.0
Node Description, Node Number, Nastran Node ID Number, Location Coordinates (X, Y, Z)

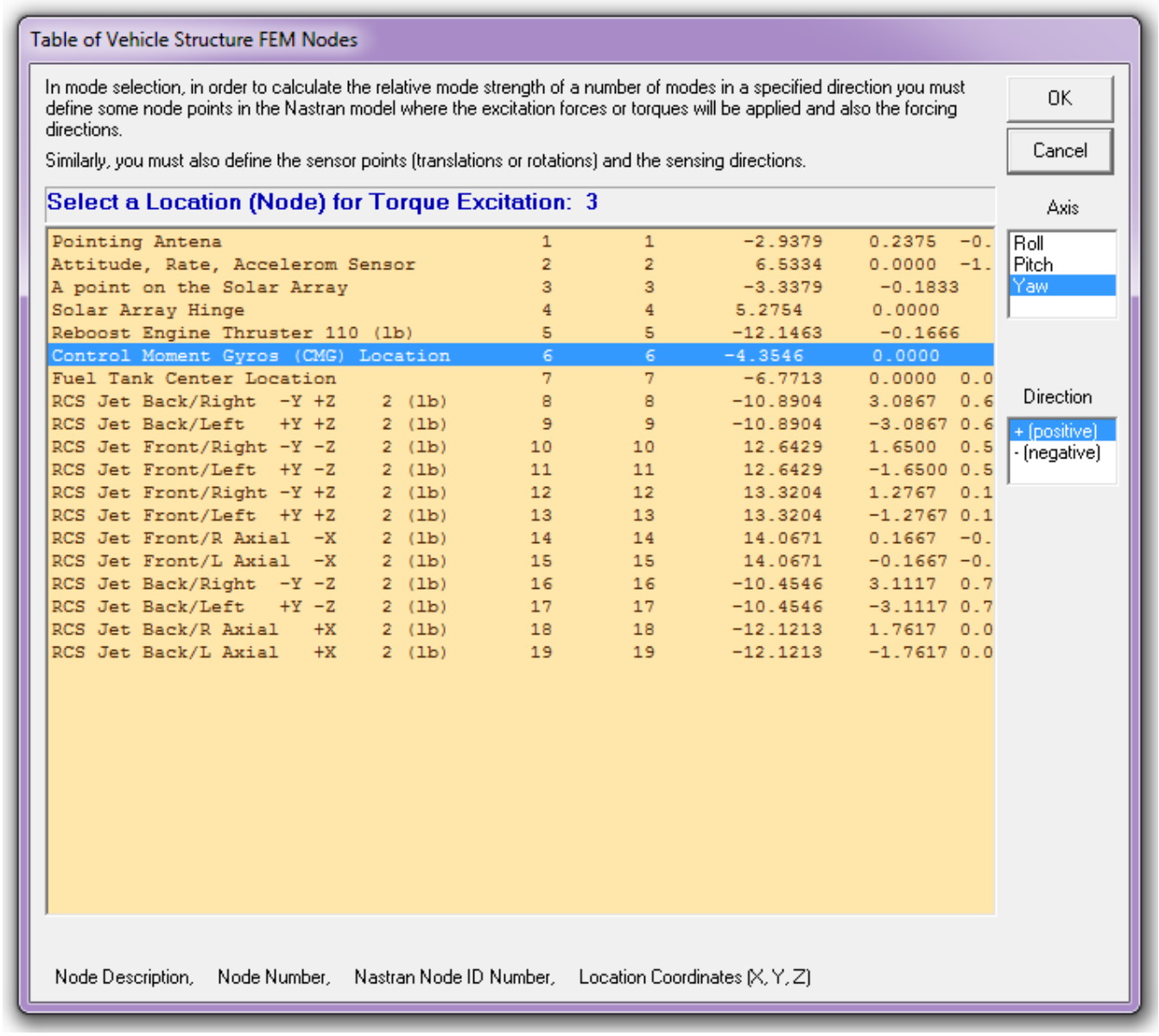


Table of Vehicle Structure FEM Nodes
In mode selection, in order to calculate the relative mode strength of a number of modes in a specified direction you must define some node points in the Nastran model where the excitation forces or torques will be applied and also the forcing directions.
Similarly, you must also define the sensor points (translations or rotations) and the sensing directions.
OK
Cancel
Select a Location (Node) for Torque Excitation: 3
Axis
Roll
Pitch
Yaw
Direction
+ (positive)
- (negative)
Pointing Antena 1 1 -2.9379 0.2375 -0.
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.0
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.6
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.6
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.5
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.5
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.1
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.1
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.7
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.7
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.0
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.0
Node Description, Node Number, Nastran Node ID Number, Location Coordinates (X, Y, Z)

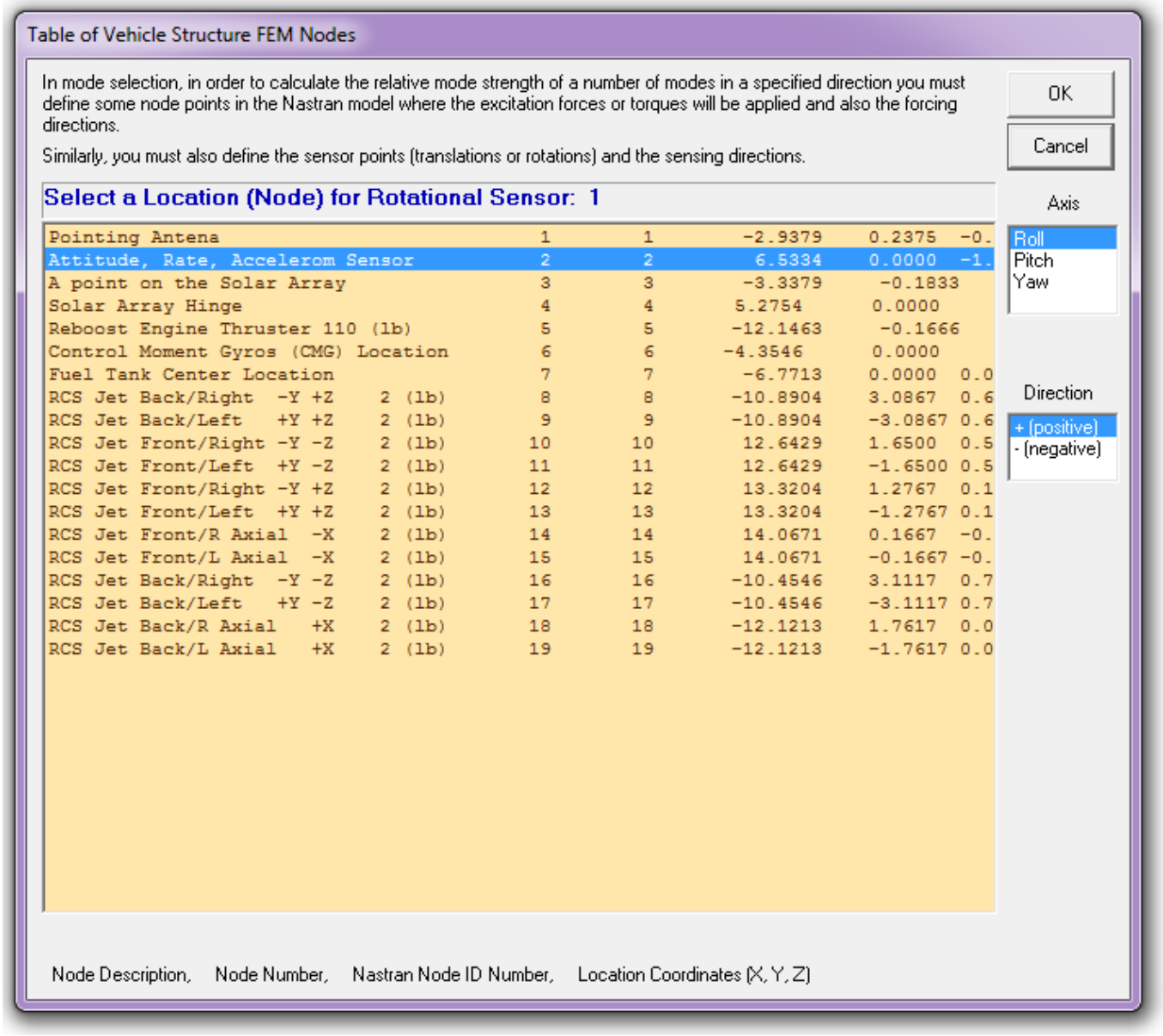
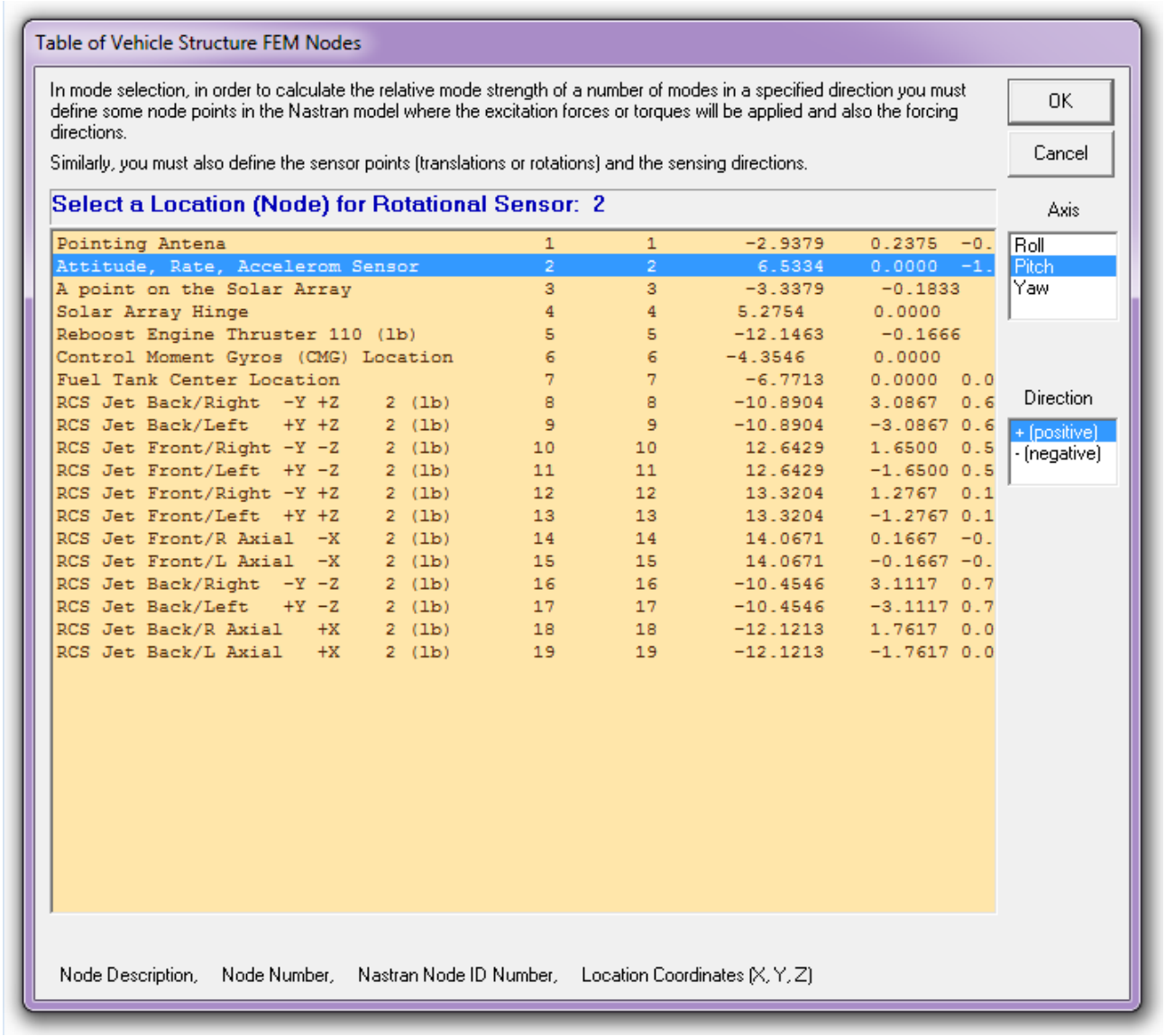
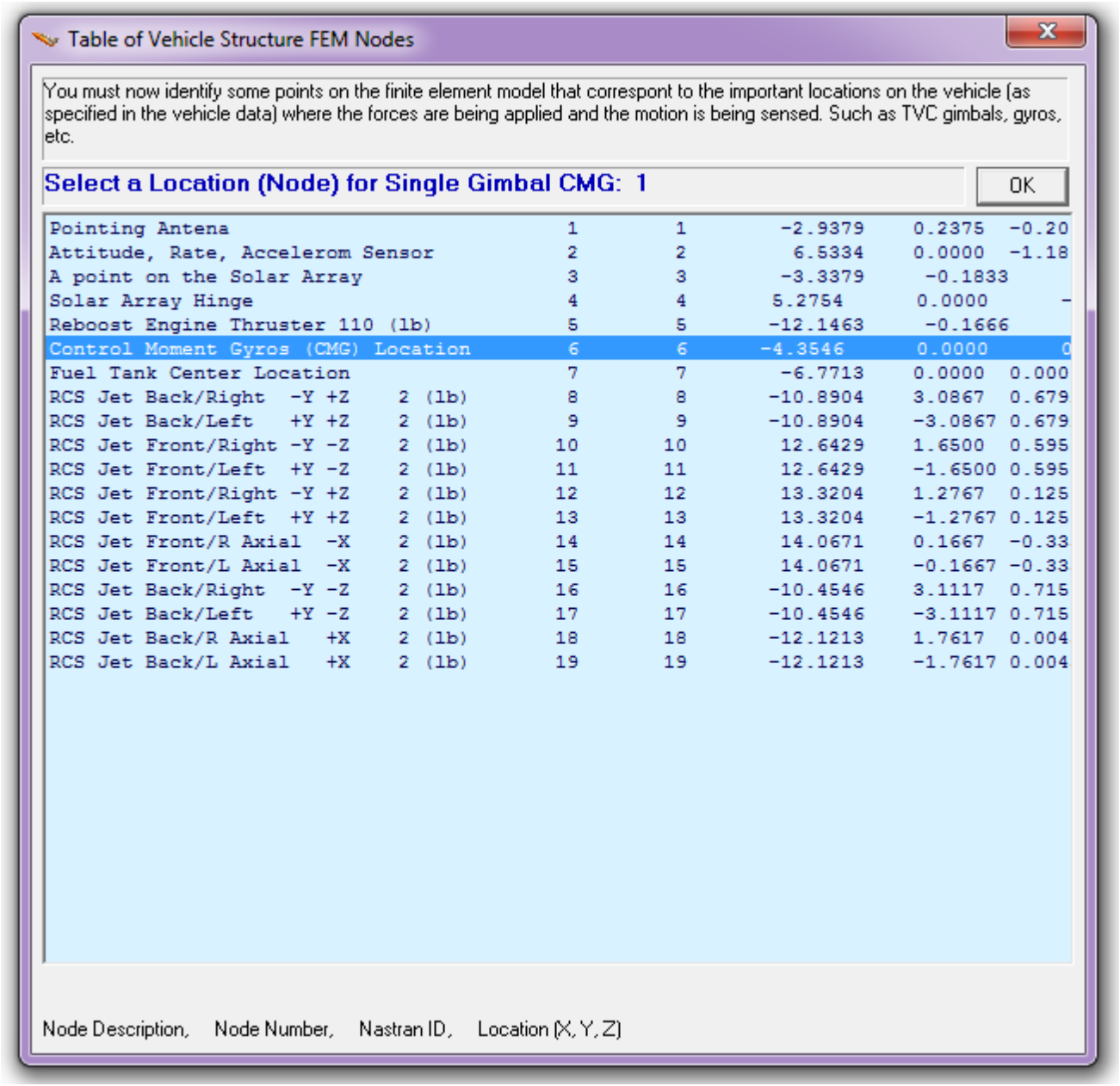


Table of Vehicle Structure FEM Nodes
In mode selection, in order to calculate the relative mode strength of a number of modes in a specified direction you must define some node points in the Nastran model where the excitation forces or torques will be applied and also the forcing directions.
Similarly, you must also define the sensor points (translations or rotations) and the sensing directions.
OK
Cancel
Select a Location (Node) for Rotational Sensor: 1
Axis
Roll
Pitch
Yaw
Direction
+ (positive)
- (negative)
Pointing Antena 1 1 -2.9379 0.2375 -0.
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.0
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.6
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.6
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.5
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.5
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.1
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.1
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.7
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.7
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.0
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.0
Node Description, Node Number, Nastran Node ID Number, Location Coordinates (X, Y, Z)



At this point the excitation and sensor points and directions for the mode selection process have been defined. The modal strength for each mode is calculated based on the values of the modal slopes at the torque locations and at the rotational measurements. When the mode shape magnitudes at the excitation and sensor points are high at a mode frequency it implies that this mode has a strong contribution to the overall structure flexibility. The mode selection program computes the mode strength at each mode frequency and saves it in file "*Modsel.Msl*".

The mode selection process, however, is not finished yet because the program needs additional info before allowing the user to select which modes to retain from the big modal data file. Using similar menus that display the nodes table (different light-blue color), the user must select four nodes from the finite element model that correspond to the four CMGs which are defined in the vehicle input data. Node #6 is selected for all four CMGs.



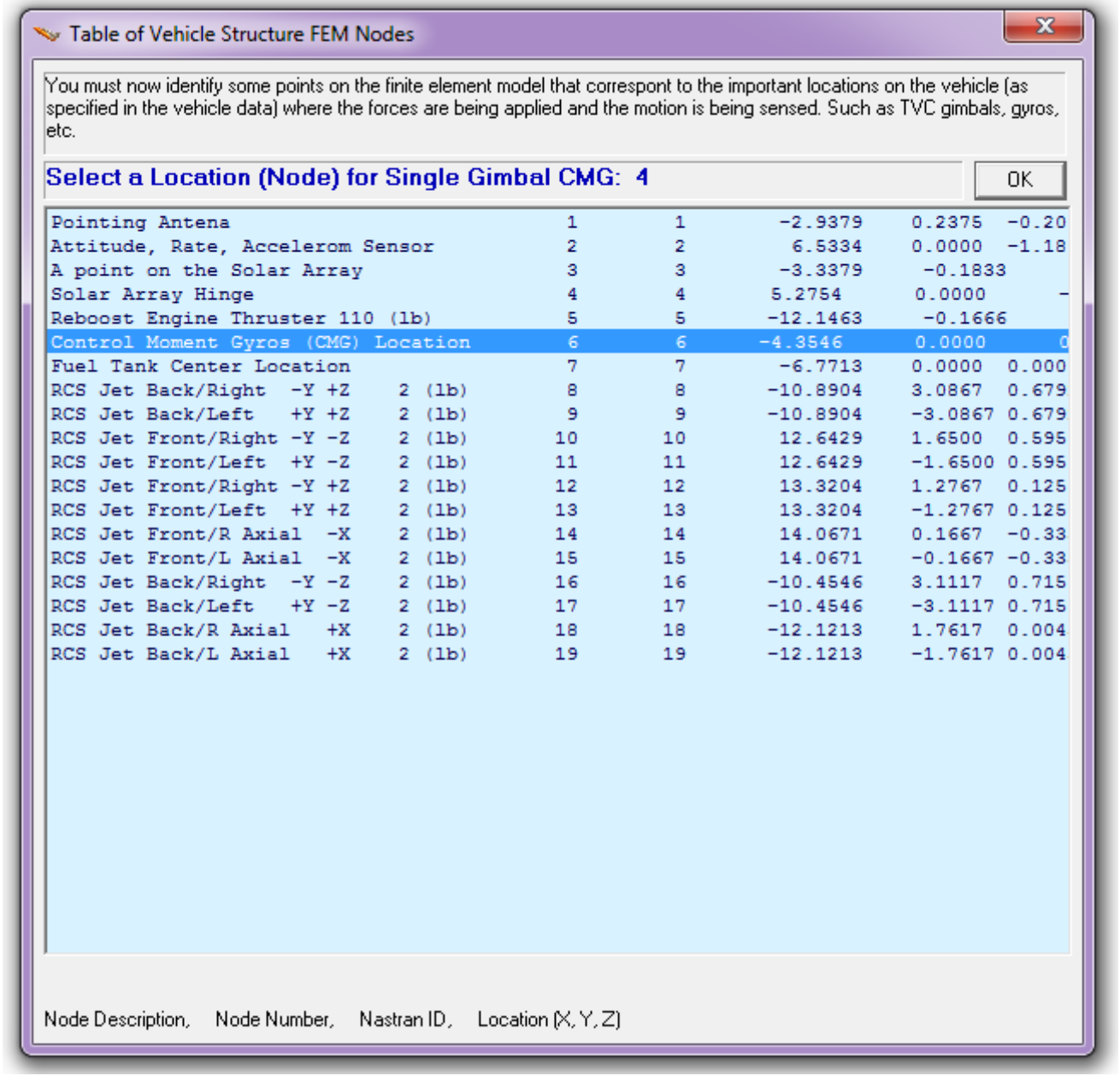


Table of Vehicle Structure FEM Nodes
You must now identify some points on the finite element model that correspont to the important locations on the vehicle (as specified in the vehicle data) where the forces are being applied and the motion is being sensed. Such as TVC gimbals, gyros, etc.
Select a Location (Node) for Single Gimbal CMG: 4
OK
Pointing Antena 1 1 -2.9379 0.2375 -0.20
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.18
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000 -
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000 0
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.000
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.679
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.679
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.595
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.595
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.125
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.125
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.33
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.33
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.715
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.715
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.004
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.004
Node Description, Node Number, Nastran ID, Location (X, Y, Z)

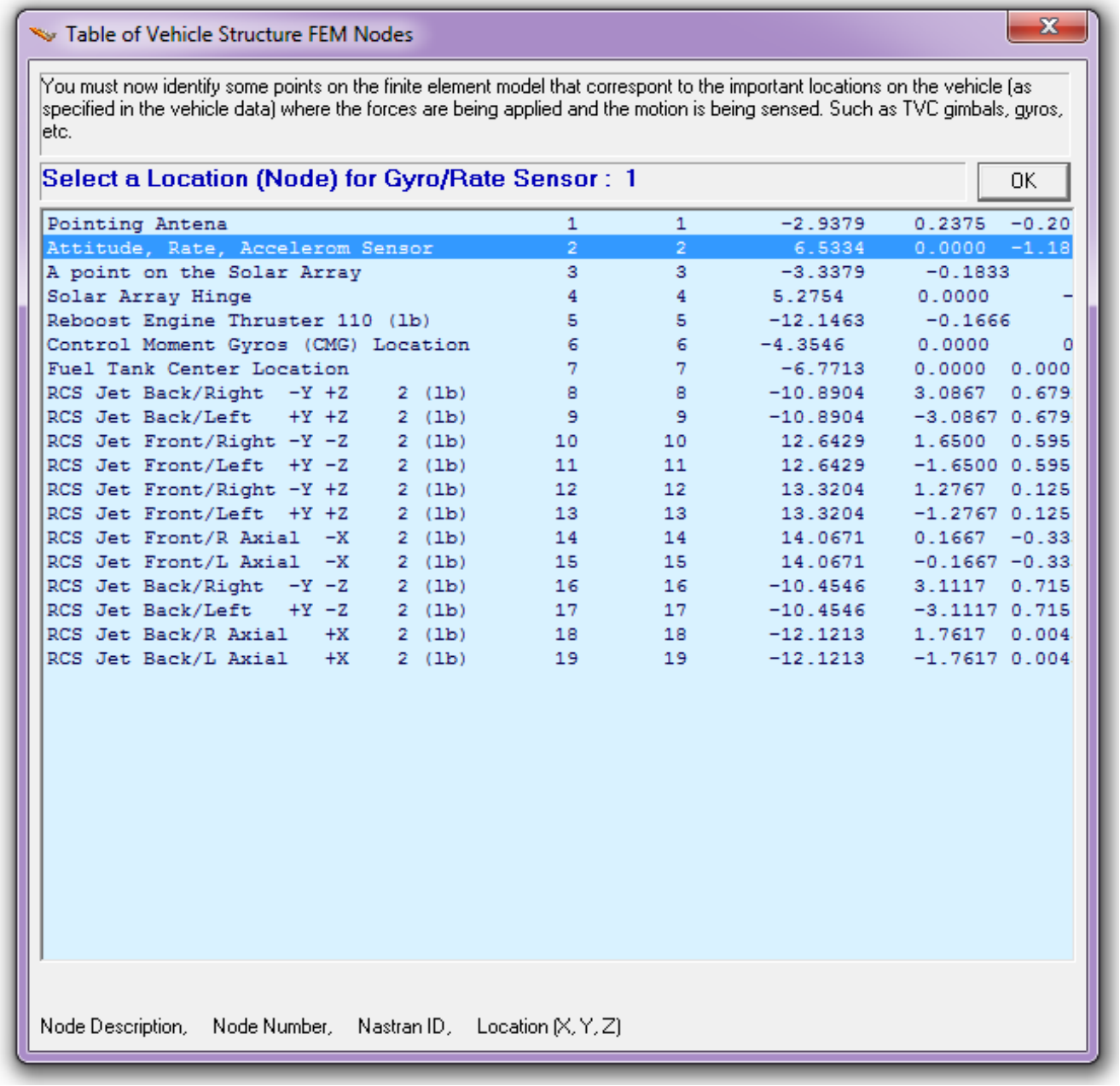
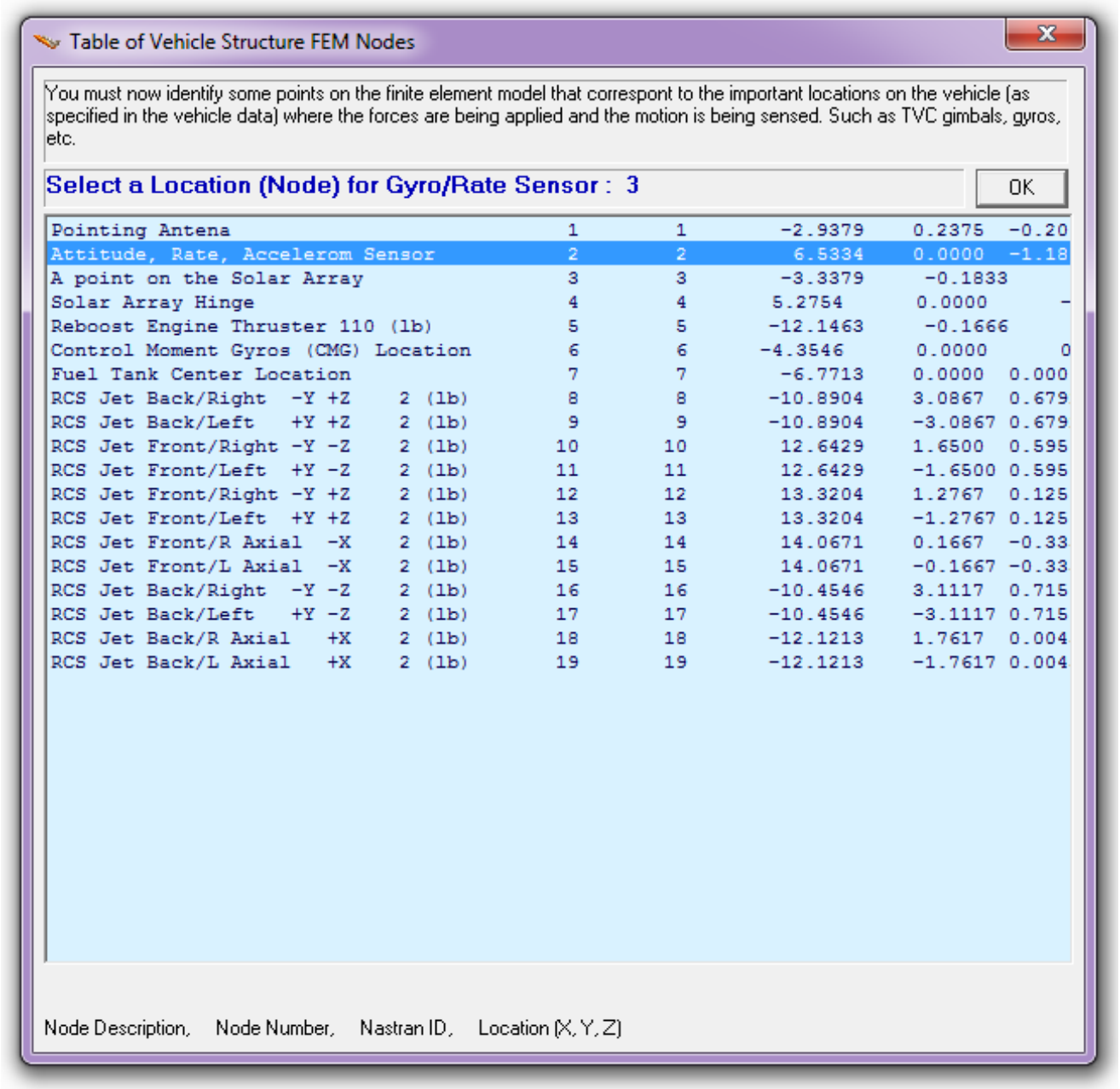


Table of Vehicle Structure FEM Nodes
You must now identify some points on the finite element model that correspont to the important locations on the vehicle (as specified in the vehicle data) where the forces are being applied and the motion is being sensed. Such as TVC gimbals, gyros, etc.
Select a Location (Node) for Gyro/Rate Sensor : 1
OK
Pointing Antena 1 1 -2.9379 0.2375 -0.20
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.18
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000 -
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000 0
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.000
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.679
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.679
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.595
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.595
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.125
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.125
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.33
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.33
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.715
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.715
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.004
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.004
Node Description, Node Number, Nastran ID, Location (X, Y, Z)

The user also selects 9 nodes (locations) for the 9 rotational sensors which are defined in the vehicle data, title "*Flexible Agile Spacecraft with 4 SG-CMG*". That is, three rotations at node #2, three rotational rates at node #2, and three rotational rates at node #1.



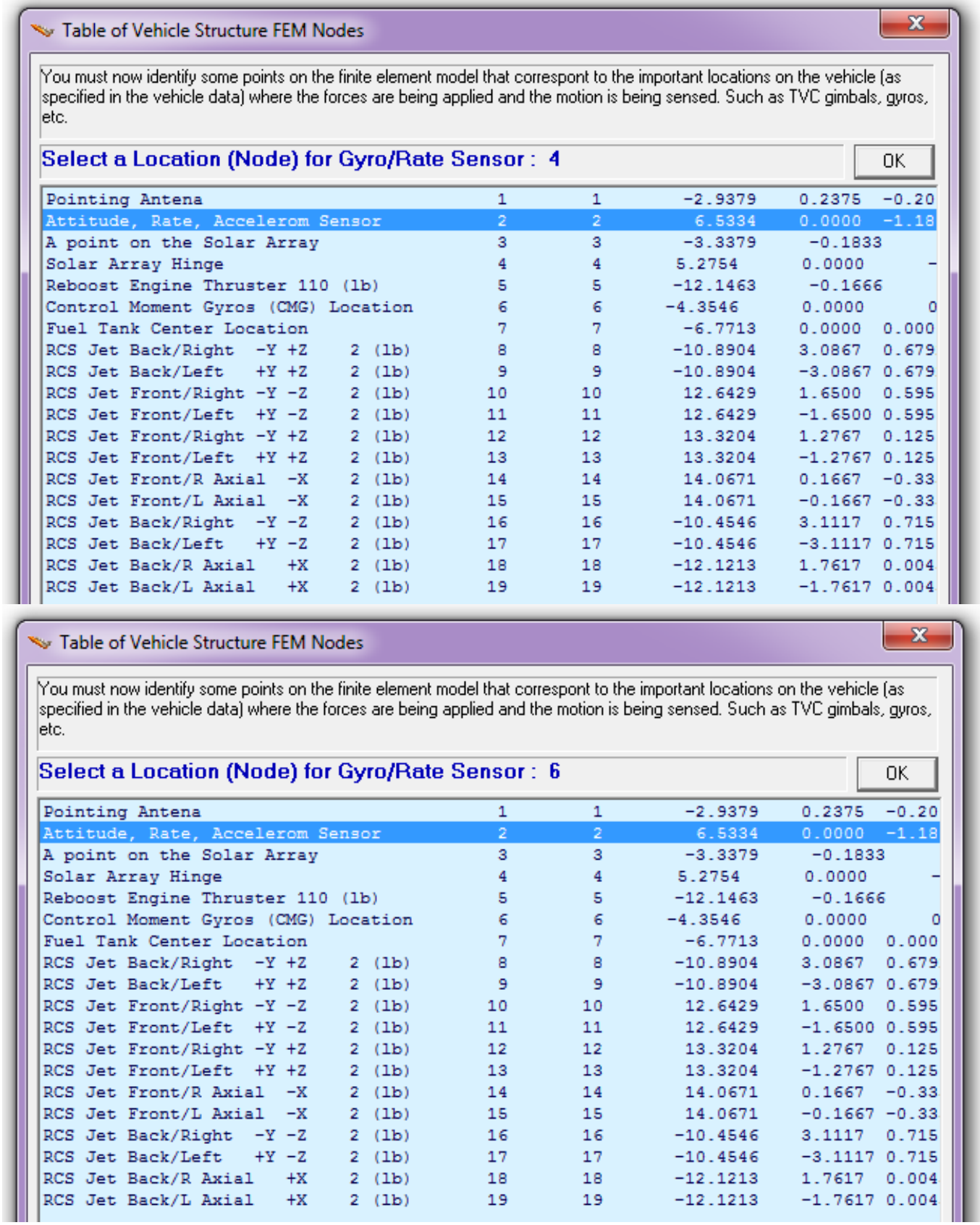


Table of Vehicle Structure FEM Nodes
You must now identify some points on the finite element model that correspont to the important locations on the vehicle (as specified in the vehicle data) where the forces are being applied and the motion is being sensed. Such as TVC gimbals, gyros, etc.
Select a Location (Node) for Gyro/Rate Sensor : 4
OK
Pointing Antena 1 1 -2.9379 0.2375 -0.20
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.18
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000 -
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000 0
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.000
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.679
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.679
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.595
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.595
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.125
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.125
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.33
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.33
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.715
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.715
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.004
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.004
Table of Vehicle Structure FEM Nodes
You must now identify some points on the finite element model that correspont to the important locations on the vehicle (as specified in the vehicle data) where the forces are being applied and the motion is being sensed. Such as TVC gimbals, gyros, etc.
Select a Location (Node) for Gyro/Rate Sensor : 6
OK
Pointing Antena 1 1 -2.9379 0.2375 -0.20
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.18
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000 -
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000 0
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.000
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.679
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.679
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.595
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.595
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.125
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.125
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.33
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.33
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.715
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.715
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.004
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.004

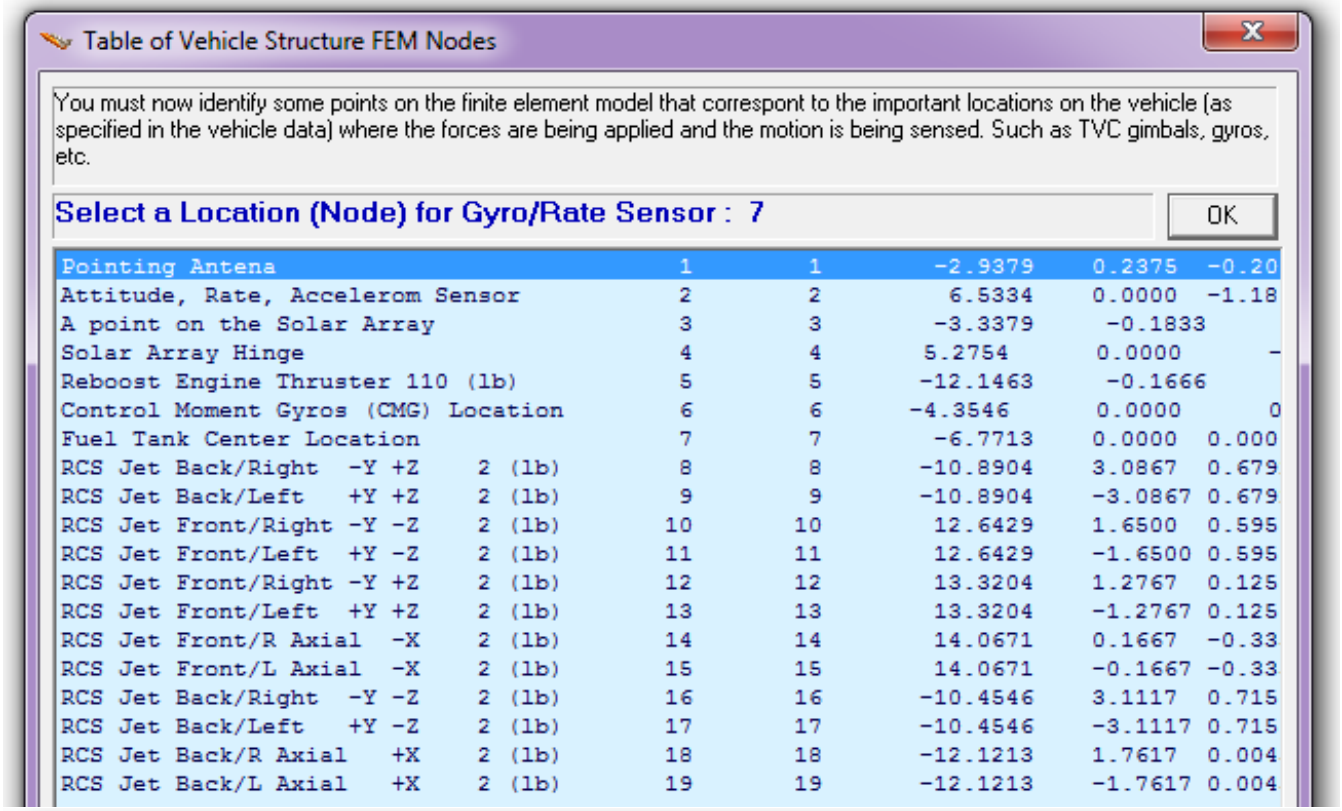


Table of Vehicle Structure FEM Nodes
You must now identify some points on the finite element model that correspont to the important locations on the vehicle (as specified in the vehicle data) where the forces are being applied and the motion is being sensed. Such as TVC gimbals, gyros, etc.
Select a Location (Node) for Gyro/Rate Sensor : 7
OK
Pointing Antena 1 1 -2.9379 0.2375 -0.20
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.18
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000 -
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000 0
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.000
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.679
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.679
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.595
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.595
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.125
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.125
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.33
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.33
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.715
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.715
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.004
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.004

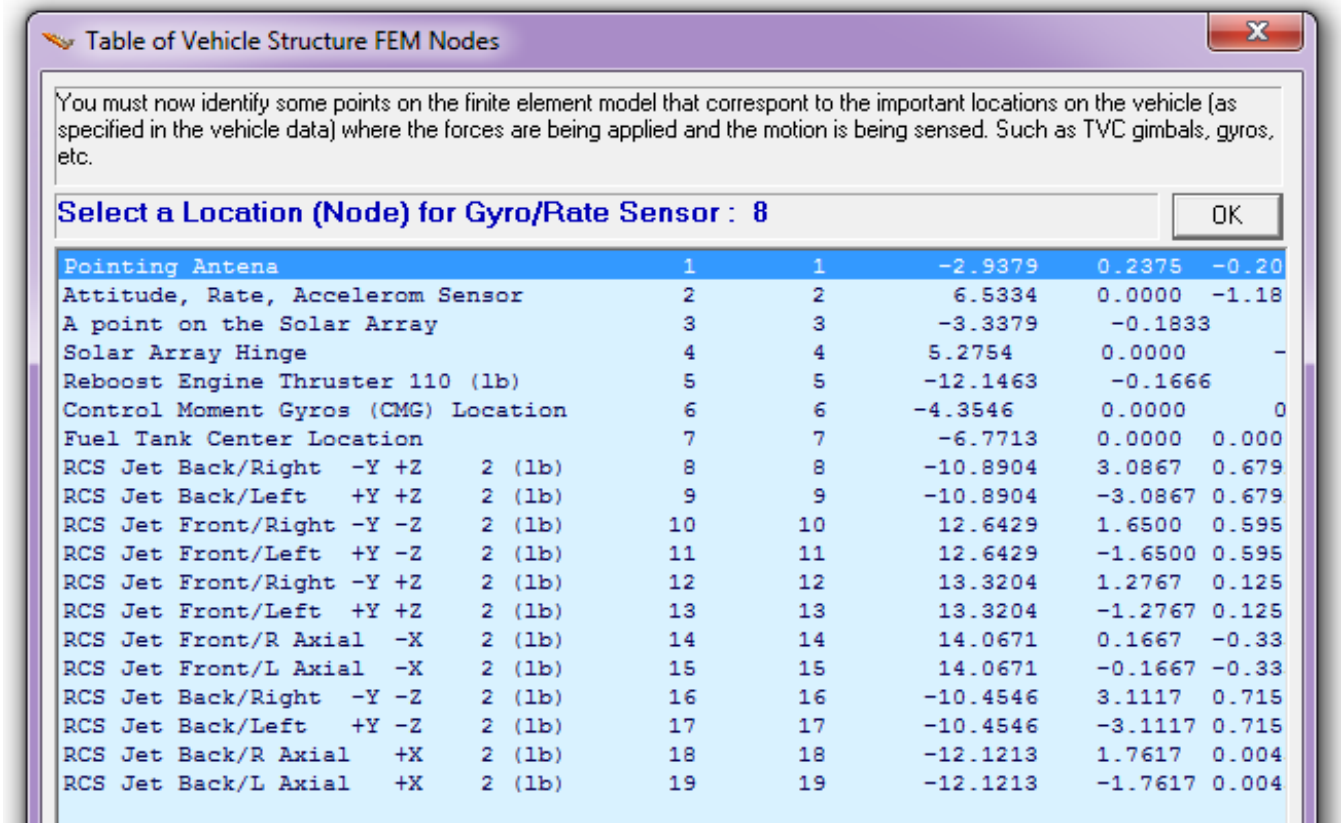
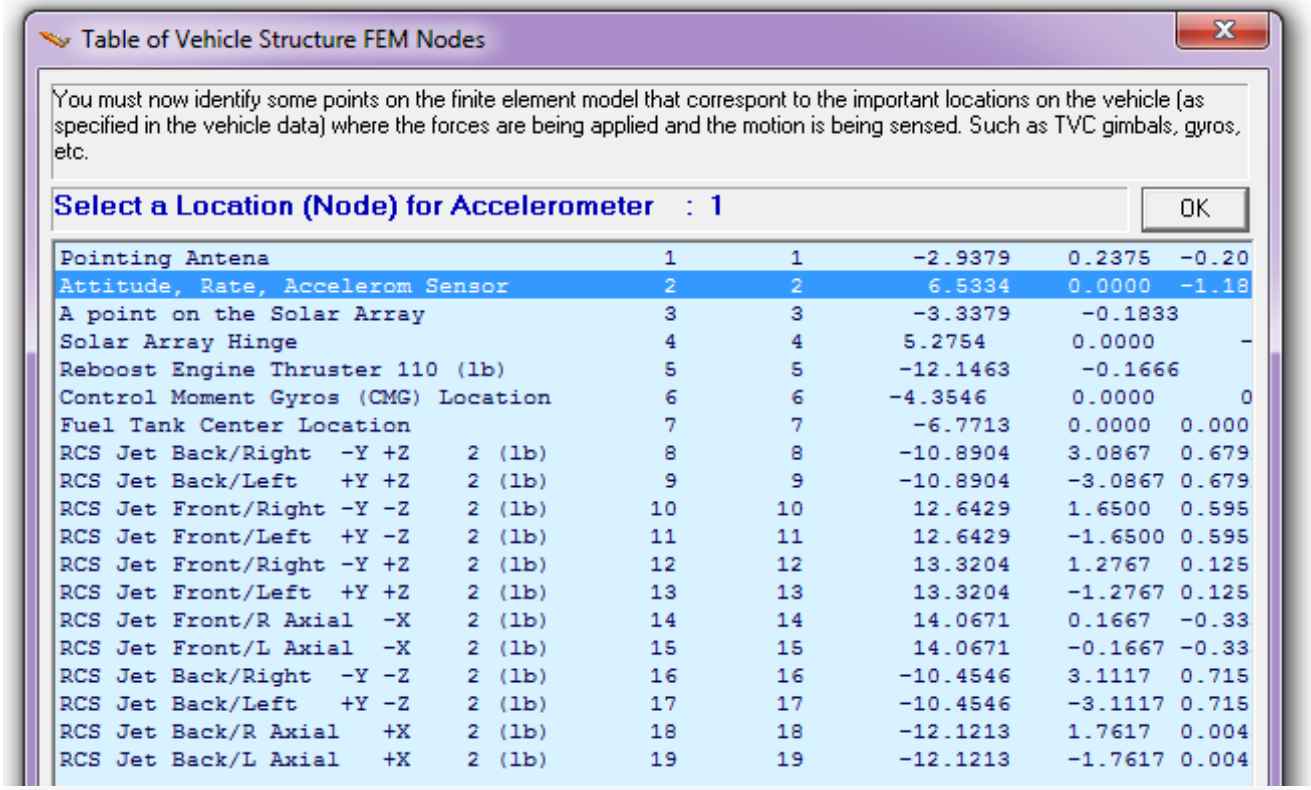
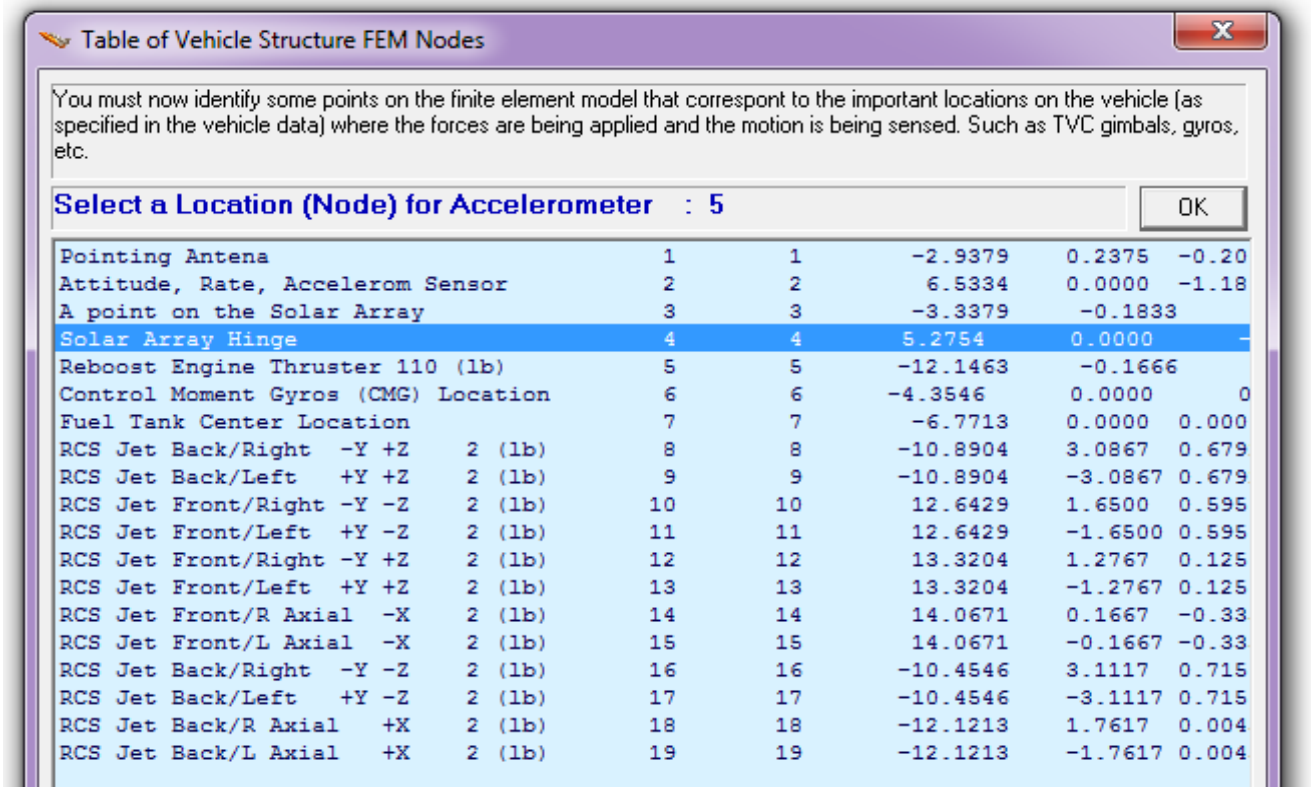
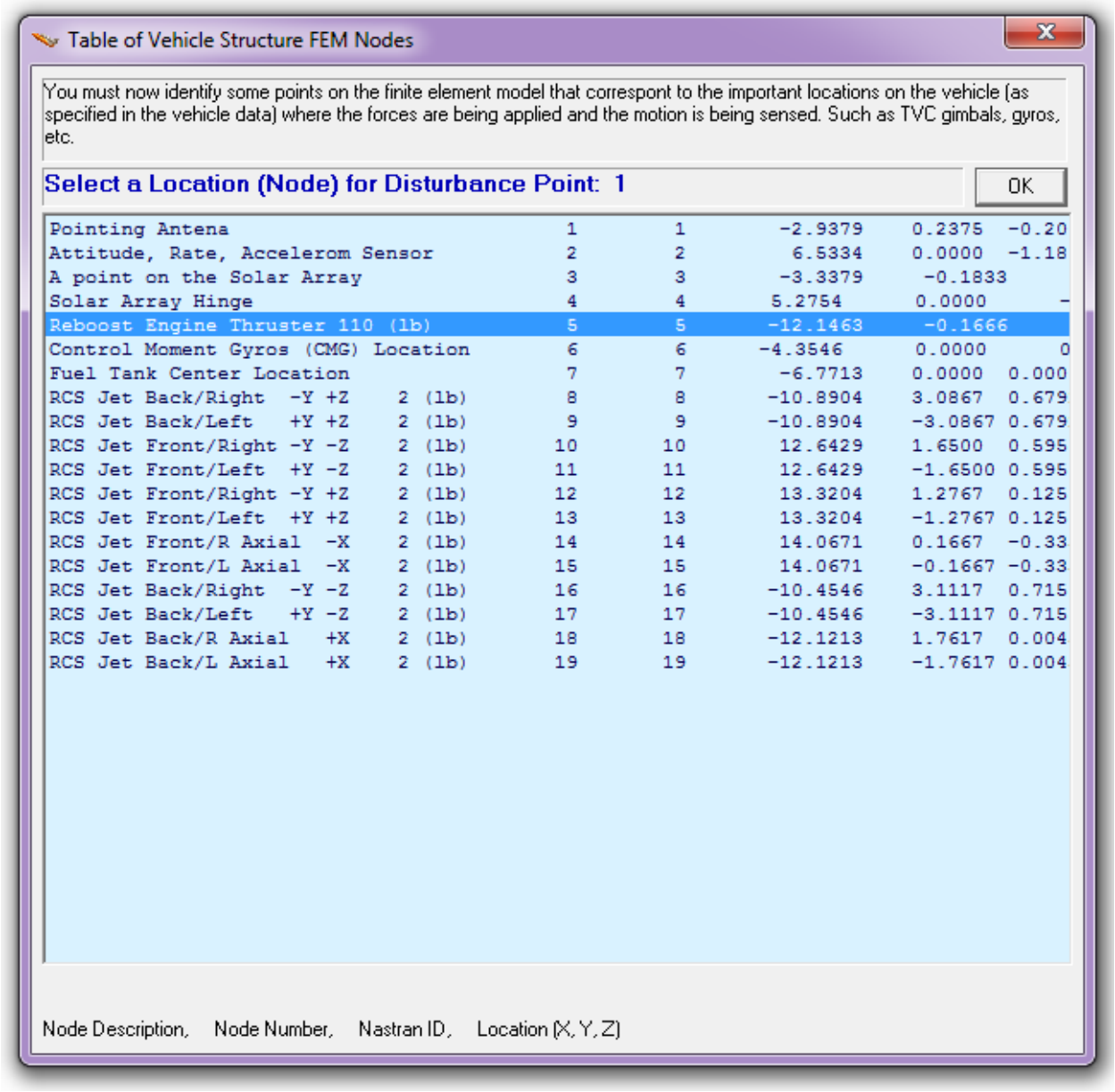


Table of Vehicle Structure FEM Nodes
You must now identify some points on the finite element model that correspont to the important locations on the vehicle (as specified in the vehicle data) where the forces are being applied and the motion is being sensed. Such as TVC gimbals, gyros, etc.
Select a Location (Node) for Gyro/Rate Sensor : 8
OK
Pointing Antena 1 1 -2.9379 0.2375 -0.20
Attitude, Rate, Accelerom Sensor 2 2 6.5334 0.0000 -1.18
A point on the Solar Array 3 3 -3.3379 -0.1833
Solar Array Hinge 4 4 5.2754 0.0000 -
Reboost Engine Thruster 110 (lb) 5 5 -12.1463 -0.1666
Control Moment Gyros (CMG) Location 6 6 -4.3546 0.0000 0
Fuel Tank Center Location 7 7 -6.7713 0.0000 0.000
RCS Jet Back/Right -Y +Z 2 (lb) 8 8 -10.8904 3.0867 0.679
RCS Jet Back/Left +Y +Z 2 (lb) 9 9 -10.8904 -3.0867 0.679
RCS Jet Front/Right -Y -Z 2 (lb) 10 10 12.6429 1.6500 0.595
RCS Jet Front/Left +Y -Z 2 (lb) 11 11 12.6429 -1.6500 0.595
RCS Jet Front/Right -Y +Z 2 (lb) 12 12 13.3204 1.2767 0.125
RCS Jet Front/Left +Y +Z 2 (lb) 13 13 13.3204 -1.2767 0.125
RCS Jet Front/R Axial -X 2 (lb) 14 14 14.0671 0.1667 -0.33
RCS Jet Front/L Axial -X 2 (lb) 15 15 14.0671 -0.1667 -0.33
RCS Jet Back/Right -Y -Z 2 (lb) 16 16 -10.4546 3.1117 0.715
RCS Jet Back/Left +Y -Z 2 (lb) 17 17 -10.4546 -3.1117 0.715
RCS Jet Back/R Axial +X 2 (lb) 18 18 -12.1213 1.7617 0.004
RCS Jet Back/L Axial +X 2 (lb) 19 19 -12.1213 -1.7617 0.004





Similarly, select nodes for the 6 accelerometers. The first 3 accelerometers are at node #2, the nav base, and the next 3 are at node #4, the solar array.

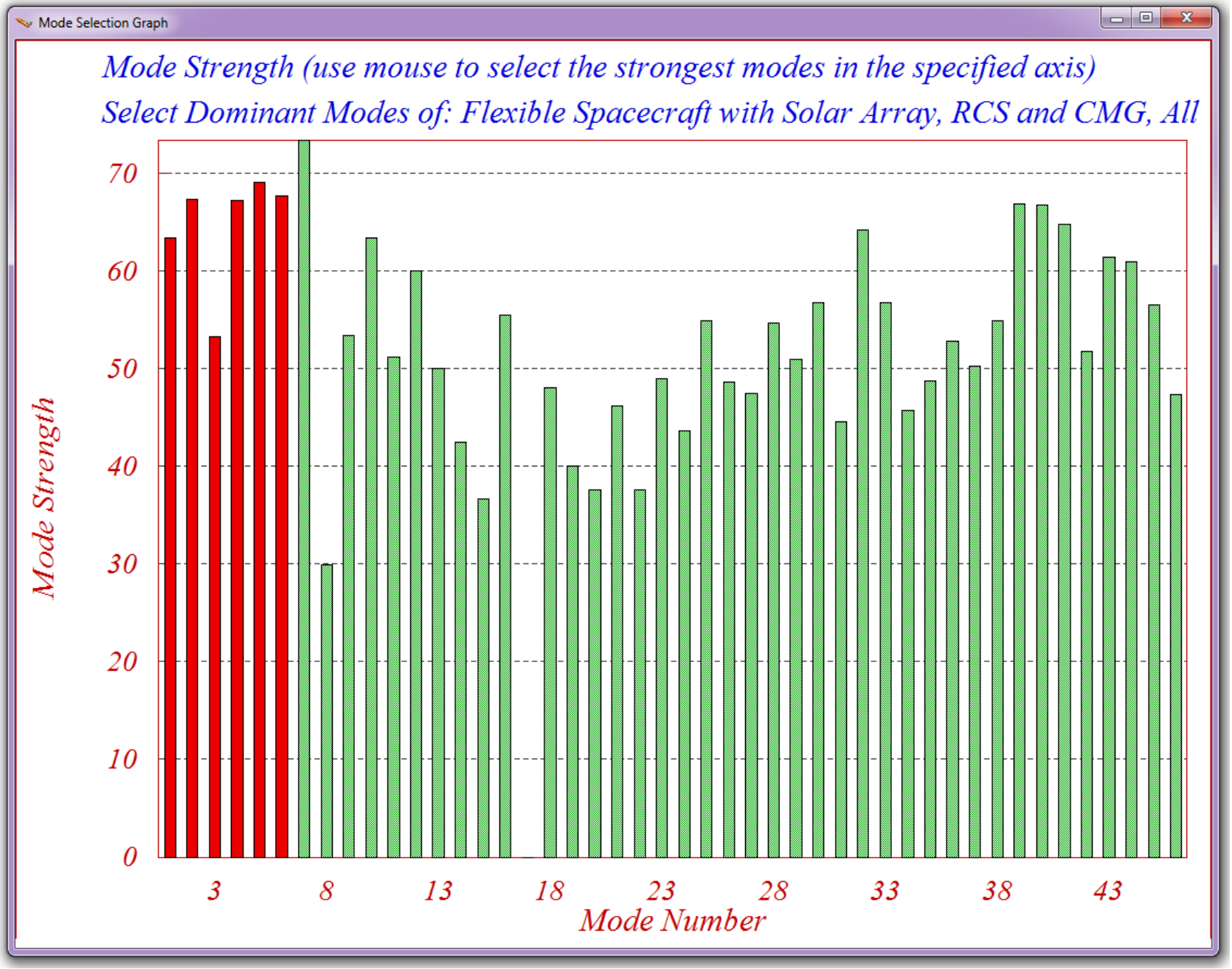


We must also select a point where we can apply the disturbance.

The program will create a smaller subset of the original modal data set and will save it at the bottom of the input data file "*FlexSc_CMG_FVP.Inp*". At the end of this process the selected set of modal data will contain only the dominant modes (between the excitation and sensor points defined), and mode shapes only at the locations which are required by the flexible spacecraft model, such as: the CMG locations, the gyros, and the accelerometers. A title will be created for the selected and rescaled set of modal data. The selected modes title is "*Flexible Agile Spacecraft with 4 SG-CMG, All Flex Modes*", similar to the original vehicle title plus a short attachment inserted at the end by means of the following dialog.

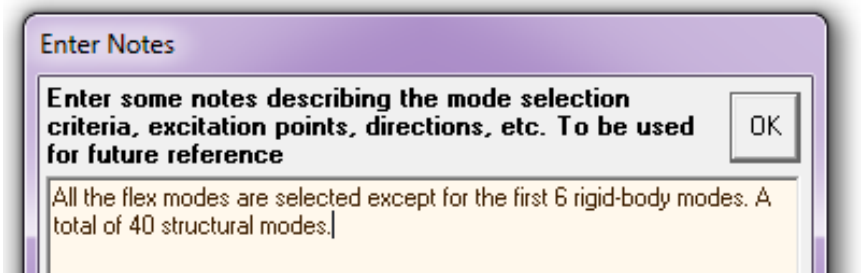
At this point the mode selection program saves the list of relative mode strengths (computed between the excitation points and directions and the sensor points and directions) in file “Modsel.Dat”. It also displays a bar-chart plot (shown in the next page) that shows the relative mode strength for each mode as a vertical bar, versus the mode number. All bars are initially red before selection. The height of each bar is logarithmically proportional to the relative mode strength. The strong modes are tall and the weak modes are short. The modal strength is a relative number adjusted with respect to the minimum and maximum modal strengths. The user selects some of the strongest modes from the chart by pointing the mouse cursor at the bar and clicking the mouse to select it. The modes change color from red to green when they are selected.

Notice that the first six modes must not be selected because they are rigid-body modes, and the rigid-body dynamics are already included in the vehicle model. We select all of the remaining flex modes from mode #7 to mode #46, a total of 40 flex modes, and press the "enter" button to complete the mode selection. They are the same flex modes that we selected in previous sections using the Flex Spacecraft Modeling program.


Mode Selection Graph
Mode Strength (use mouse to select the strongest modes in the specified axis)
Select Dominant Modes of: Flexible Spacecraft with Solar Array, RCS and CMG, All
Mode Strength
0
10
20
30
40
50
60
70
3
8
13
18
23
28
33
38
43
Mode Number

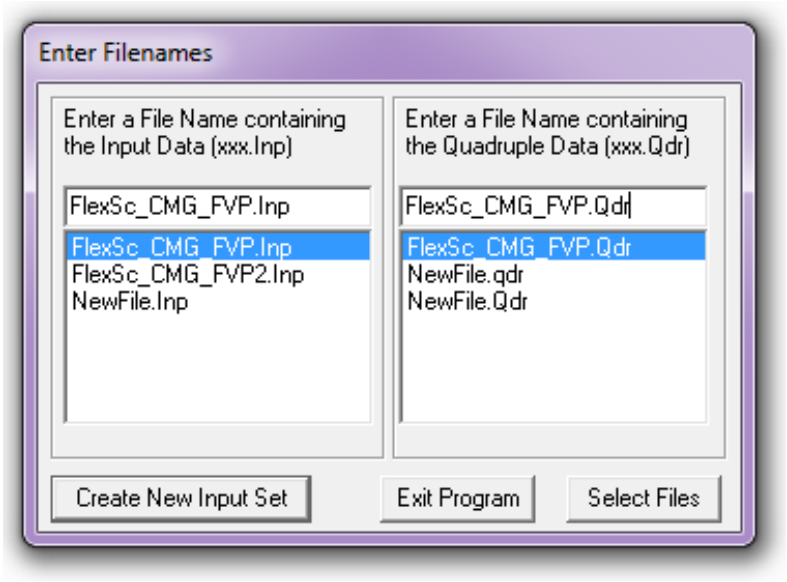
The final step before exiting mode selection is to complete the dialog below where the user enters some reference notes regarding the mode selection process. Describing, for example, what type of modes were selected, and the conditions of mode selection, excitation points, measurement points, directions, etc. This information will be included as comments below the title in the selected modes set, which is saved in the input data file "*FlexSc_CMG_FVP.Inp*".

The title of the selected modes set is "*Flexible Agile Spacecraft with 4 SG-CMG, All Flex Modes*". It contains the frequencies and the rescaled mode shapes of the selected modes at specific spacecraft locations. This title should also appear at the bottom of the spacecraft input data set "*Flexible Agile Spacecraft with 4 SG-CMG*", below the number of flex modes. This will associate the selected modes with the spacecraft input data when the VFP processes the data.

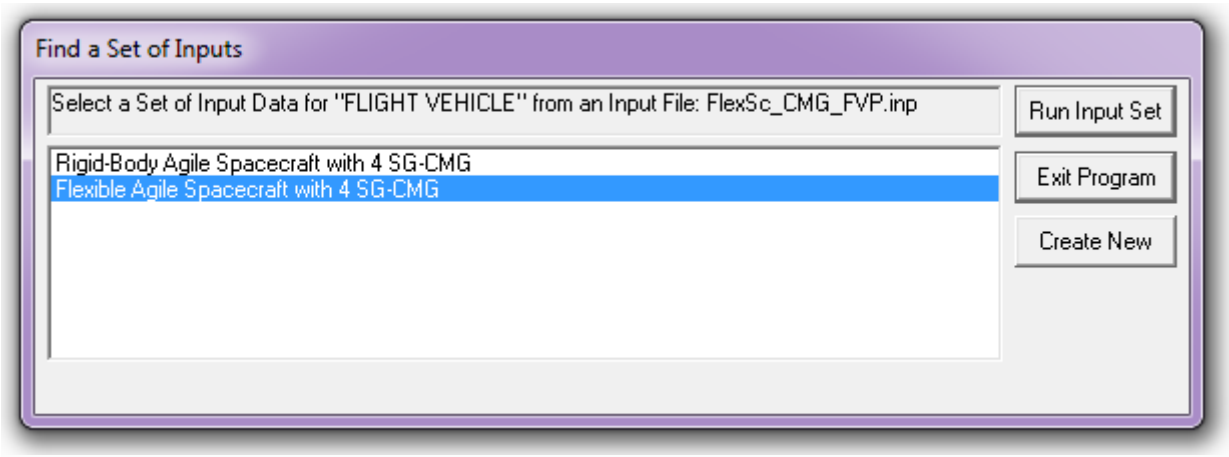


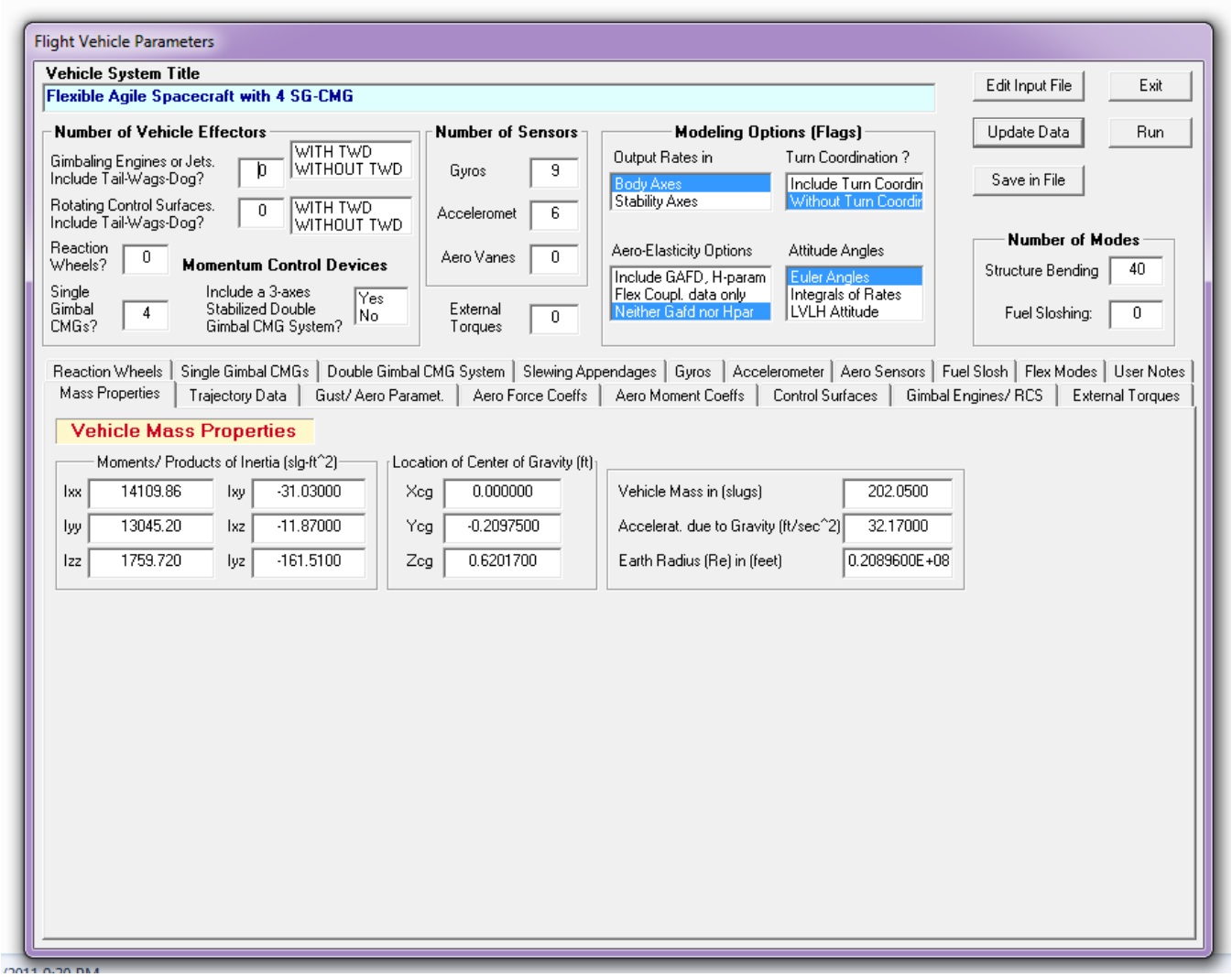
Conclusion

Having created both, the spacecraft data, and the selected flex modes we may now create the flexible spacecraft state-space system. Start the Flixan program and select the directory "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(e) 4SGCMG using FVP* ". Then go to "*Analysis Tools*", "*Flight Vehicle/ Spacecraft Modeling Tools*", and "*Flight Vehicle State-Space*". Using the following menu select the input file name "*FlexSc_CMG_FVP.Inp*", and the systems file name "*FlexSc_CMG_FVP.Qdr*", and click on "*Select Files*".



The following menu displays the titles of the two sets of spacecraft data which are saved in file "*FlexSc_CMG_FVP.Inp*". There is a rigid-body and a flexible model. We select the second title and click on "*Run Input Set*". The following dialog displays the spacecraft data which are read from the input file. We do not change anything, but click on the "*Run*" button which creates the spacecraft model in file "*FlexSc_CMG_FVP.Qdr*".





The Simulation Model

The simulation model used in this analysis is " *Lin_Flex_Sim.mdl* ", shown in Figure 3.5.1. It is similar to the non-linear model shown in the previous section but is missing all the non-linearities in the attitude control system and in the spacecraft dynamics. It uses Euler angles for attitude instead of quaternion. The integrators in the PID are "on". It is not intended to be used for large angle maneuvers but for analyzing stability and performance at nominal operating conditions, such as, the initial condition when the gimbal angles are at zero and the CMG array momentum is zero.

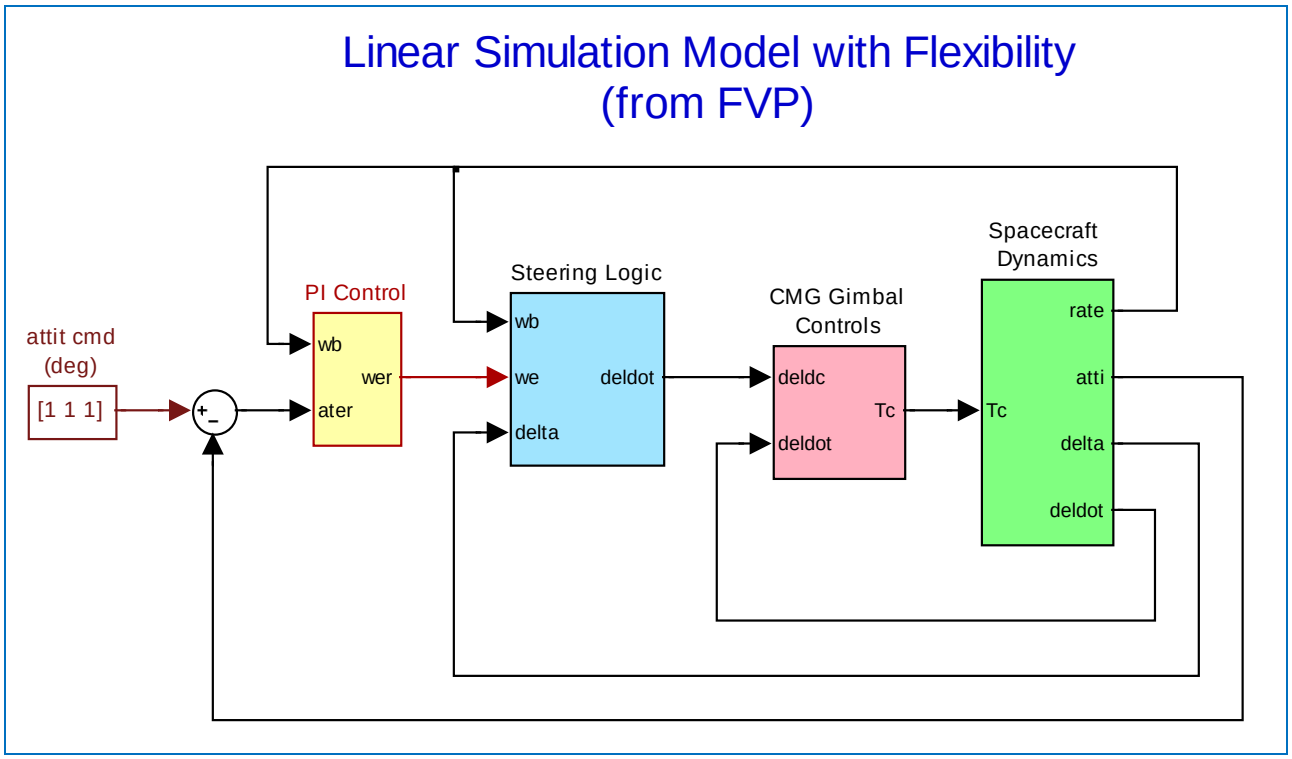
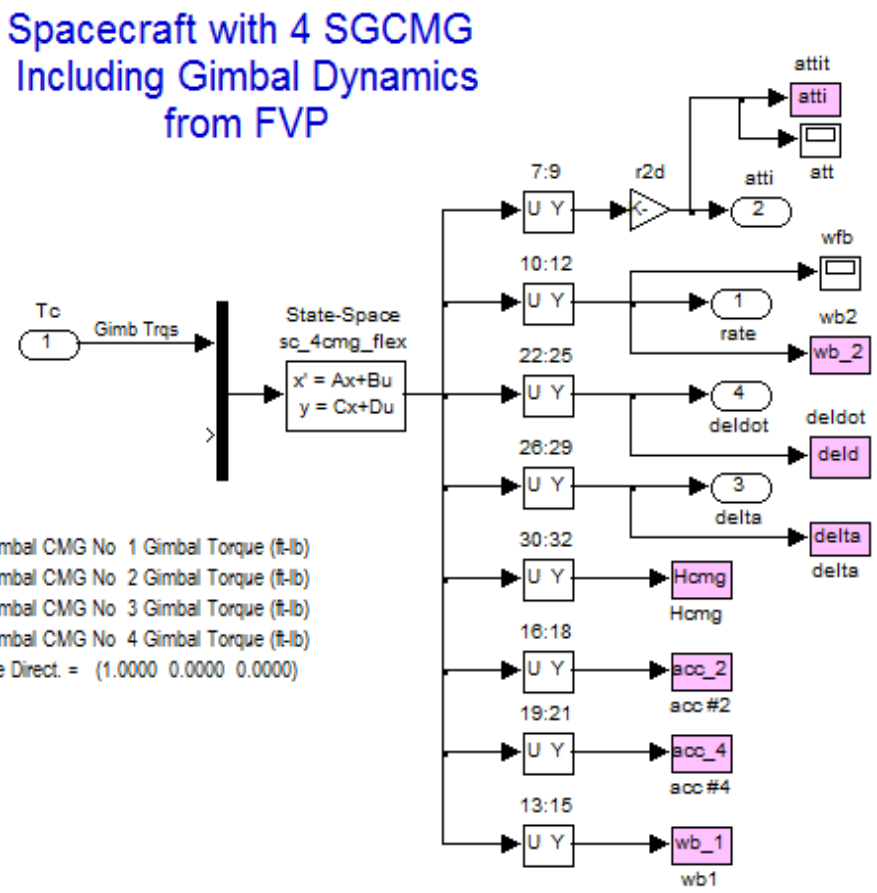


Figure 3.5.1 Linear Simulation Model "Lin_Flex_Sim.mdl"

The spacecraft plus CMG dynamics (green block) contains the state-space system from file "*sc_4cmg_flex.m*", title: "*Flexible Agile Spacecraft with 4 SG-CMG*", which was created using the Flixan FVP. The state-space system is loaded into Matlab workspace by the initialization file "start.m". It implements the linearized equations described in Section (2.2) Dynamic Equations of Spacecraft with SGCMG. It is shown in detail in Figure (3.5.2). The inputs are four gimbal torques provided by the gimbal control system designed to control the CMG gimbal rates. The gimbal servo system and the steering were described earlier. The outputs are: spacecraft attitude, rates, and accelerations at different locations, including flexibility. There is also gimbal rates, gimbal angles, and CMG array momentum in body axis. The initialization file "start.m" is also loading the gimbal control system gains, the mass properties, the CMG orientation angles, gimbal directions, and momentum reference directions which are used by the steering logic.




```
% Inputs  =  5
%   1   Single Gimbal CMG No  1 Gimbal Torque (ft-lb)
%   2   Single Gimbal CMG No  2 Gimbal Torque (ft-lb)
%   3   Single Gimbal CMG No  3 Gimbal Torque (ft-lb)
%   4   Single Gimbal CMG No  4 Gimbal Torque (ft-lb)
%   5   Ext Force Direct. =  (1.0000  0.0000  0.0000)
```

```
% Outputs =  32
%   1   Roll Attitude  (phi-body) (radians)
%   2   Pitch Attitude (thet-bdy) (radians)
%   3   Yaw  Attitude  (psi-body) (radians)
%   4   Roll Rate      (p-body)   (rad/sec)
%   5   Pitch Rate     (q-body)   (rad/sec)
%   6   Yaw  Rate      (r-body)   (rad/sec)
%   7   Gyro #  1, Roll Attitude   (Body) (radians)
%   8   Gyro #  2, Pitch Attitude  (Body) (radians)
%   9   Gyro #  3, Yaw  Attitude   (Body) (radians)
%  10   Rate-Gyro #  4, Roll Rate  (Body) (rad/sec)
%  11   Rate-Gyro #  5, Pitch Rate (Body) (rad/sec)
%  12   Rate-Gyro #  6, Yaw Rate   (Body) (rad/sec)
%  13   Rate-Gyro #  7, Roll Rate  (Body) (rad/sec)
%  14   Rate-Gyro #  8, Pitch Rate (Body) (rad/sec)
%  15   Rate-Gyro #  9, Yaw Rate   (Body) (rad/sec)
%  16   Accelerom #  1, (along X), (ft/sec^2) Translat. Acceler
%  17   Accelerom #  2, (along Y), (ft/sec^2) Translat. Acceler
%  18   Accelerom #  3, (along Z), (ft/sec^2) Translat. Acceler
%  19   Accelerom #  4, (along X), (ft/sec^2) Translat. Acceler
%  20   Accelerom #  5, (along Y), (ft/sec^2) Translat. Acceler
%  21   Accelerom #  6, (along Z), (ft/sec^2) Translat. Acceler
%  22   SGCMG # 1 Gimbal Rate (rad/sec)
%  23   SGCMG # 2 Gimbal Rate (rad/sec)
%  24   SGCMG # 3 Gimbal Rate (rad/sec)
%  25   SGCMG # 4 Gimbal Rate (rad/sec)
%  26   SGCMG # 1 Gimbal Angle (radian)
%  27   SGCMG # 2 Gimbal Angle (radian)
%  28   SGCMG # 3 Gimbal Angle (radian)
%  29   SGCMG # 4 Gimbal Angle (radian)
%  30   SGCMG Array Momentum in X-axis (ft-lb-sec)
%  31   SGCMG Array Momentum in Y-axis (ft-lb-sec)
%  32   SGCMG Array Momentum in Z-axis (ft-lb-sec)
```

The attitude controller is shown in Figure (3.5.3). It has been simplified, by removing the phase-plane, plus rate and energy limiting non-linearities, to a PID suitable for small angles steady-state operations. This is when "kay" switches from one to zero at the completion of the phase-plane mode of operation. The rate feedback loop of the PID controller is in the steering block.

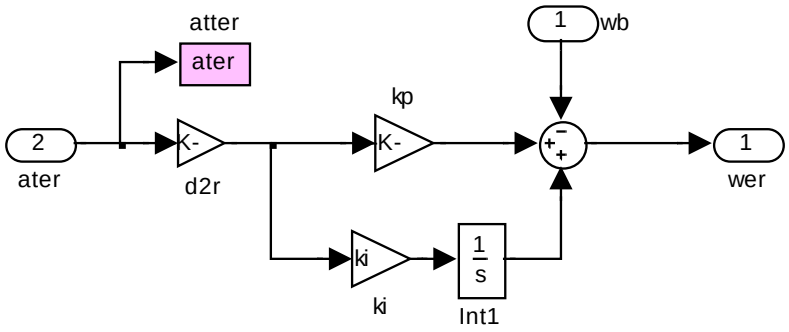


Figure 3.5.3 PI Attitude Control System

Simulation Results

The following results were obtained from the simulation model by commanding one degree attitude rotation in all 3 axes simultaneously. This model is not intended to be used for large angle maneuvers because it is missing all the non-linear controls which limit rates and torques and it will use unrealistic amounts of rates and torques. This model operates entirely in PID mode and it is intended for evaluating the spacecraft stability and performance under small angles excitation. The results show a step response time of about 2 seconds. Flexibility is more dominant in roll (blue). The attitude error in roll is affected by flexibility but it eventually decays due to modal damping. The gimbal angles respond to the flex mode excitation and so does the CMG momentum.

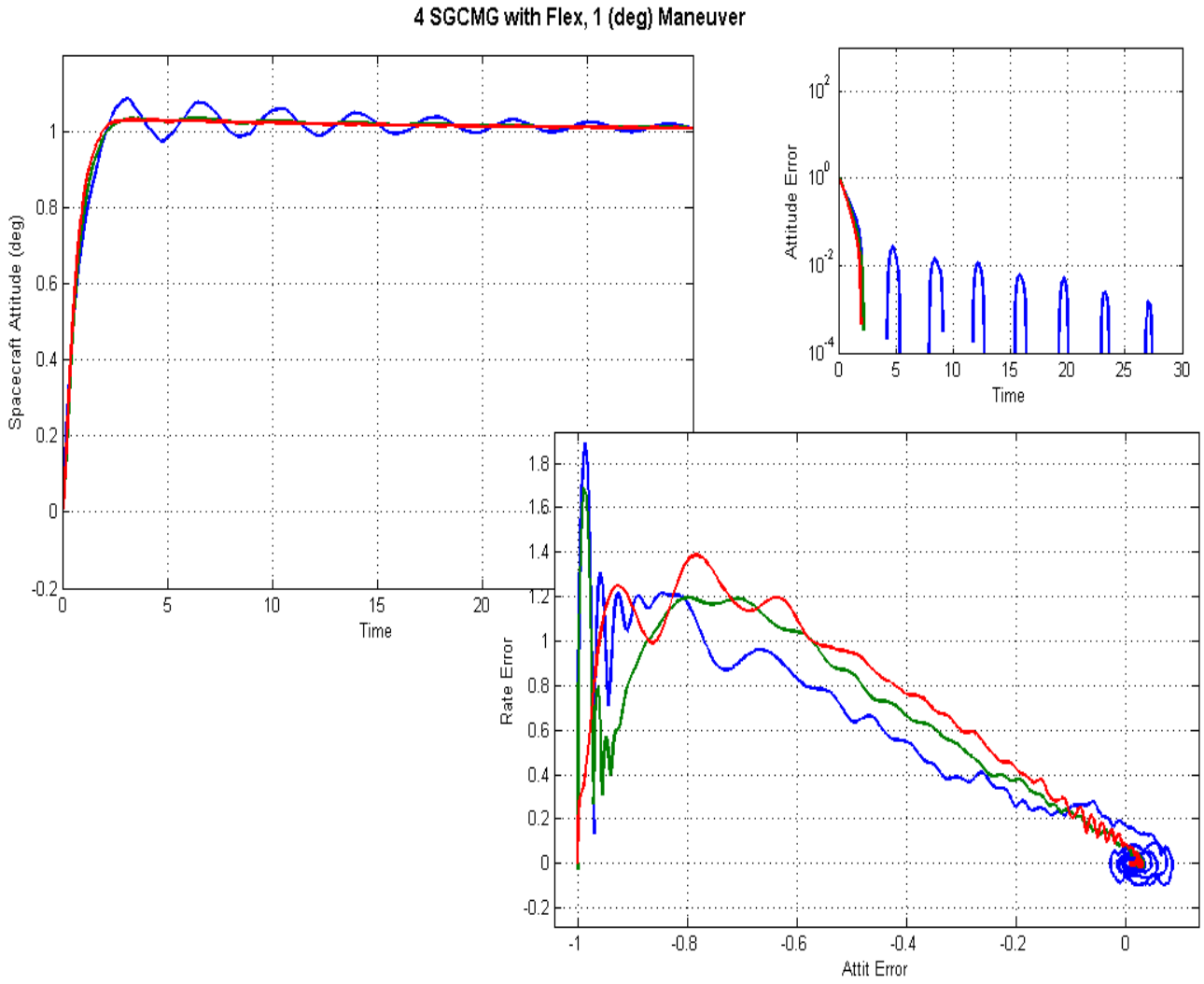


Figure 3.5.4(a) Step Response and Attitude Error

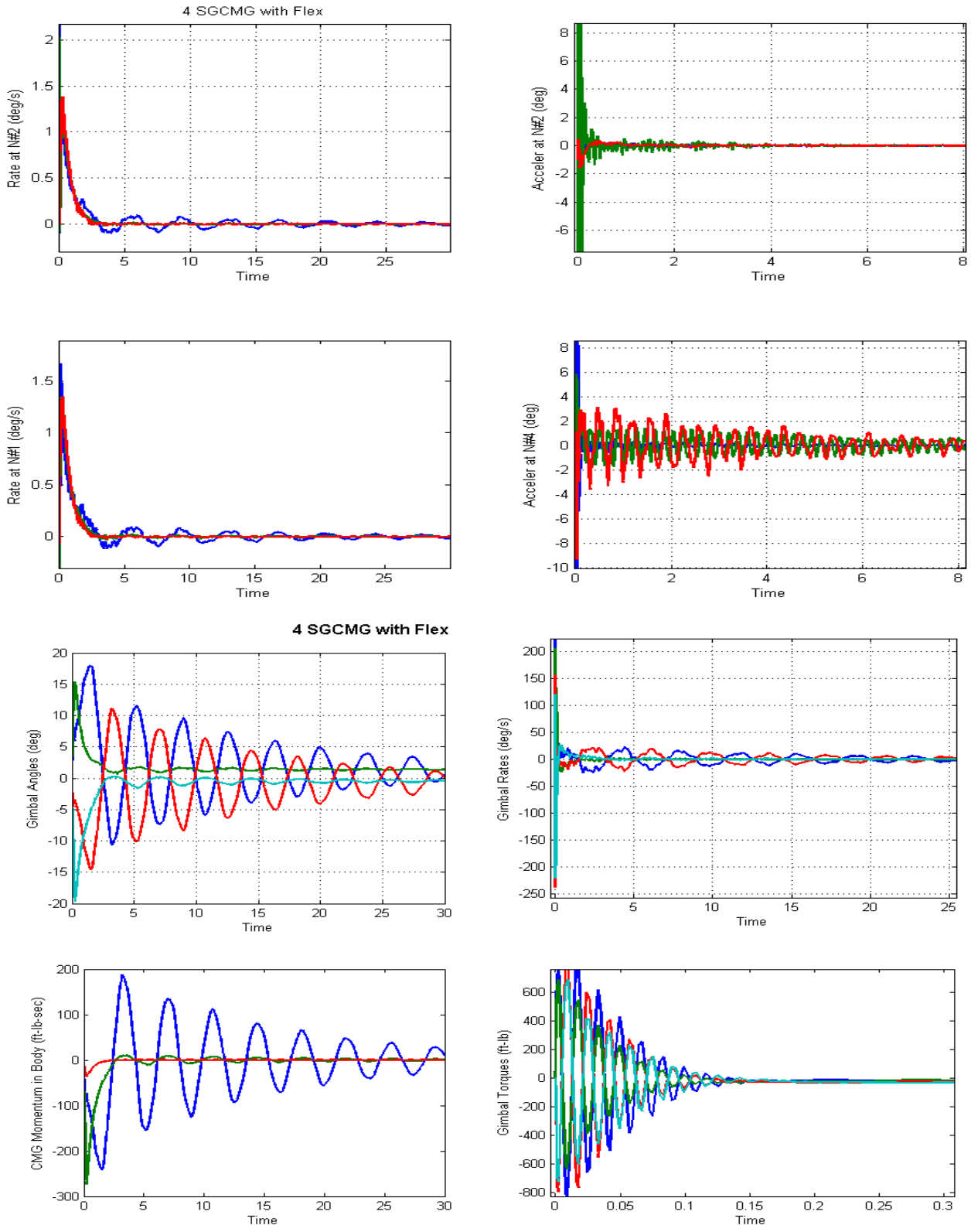


Figure 3.5.4(c) Step Response and Attitude Error

Stability Analysis

The Simulink model "*Open_Loop.mdl*", Figure 3.5.5 is used for open-loop frequency response analysis, and consists of the same subsystems as Figure 3.5.1. The control loop is broken for frequency response analysis at the PI output, which is a (roll, pitch, and yaw) rate error command to the steering logic. Only one axis is cut at a time while the other two remain closed. In the yaw analysis example shown in Figure 3.5.5, the yaw loop is opened, but the roll and pitch loops are closed. The Matlab file “freq.m” linearizes this model and calculates the frequency response between the input and the output and plots the Bode and the Nichols plots. The stability analysis plots are shown in Figure 3.5.6. Stability is measured by the phase and gain margins from the red cross. The stability margins shown for roll and pitch are poor. Some lead-lag filters were later on inserted in the PI output to improve margins (not shown here).

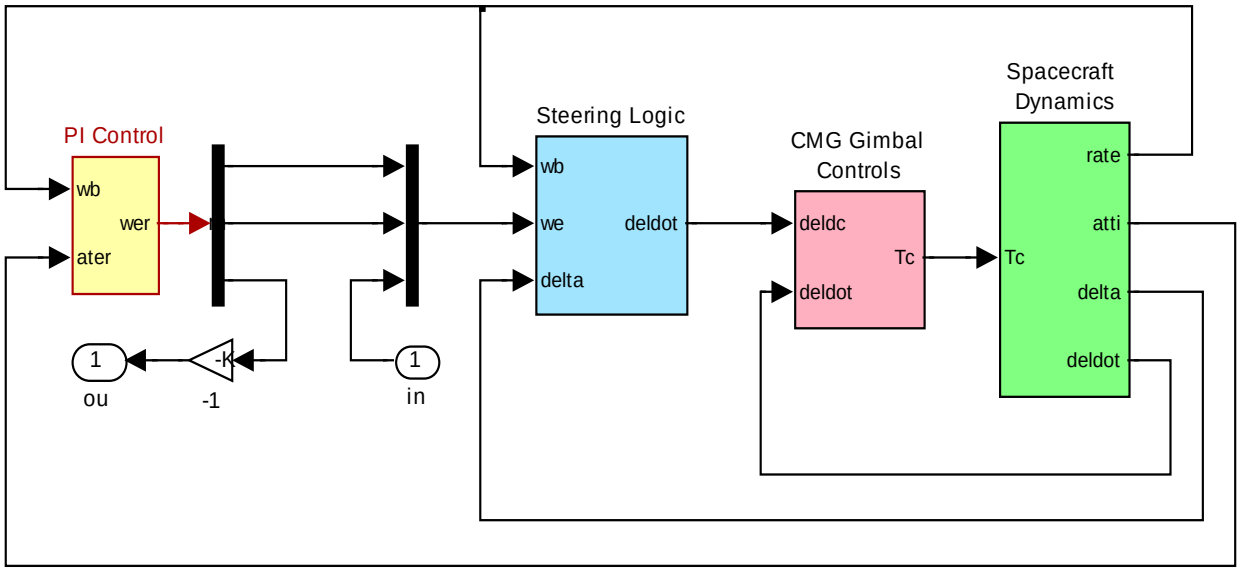
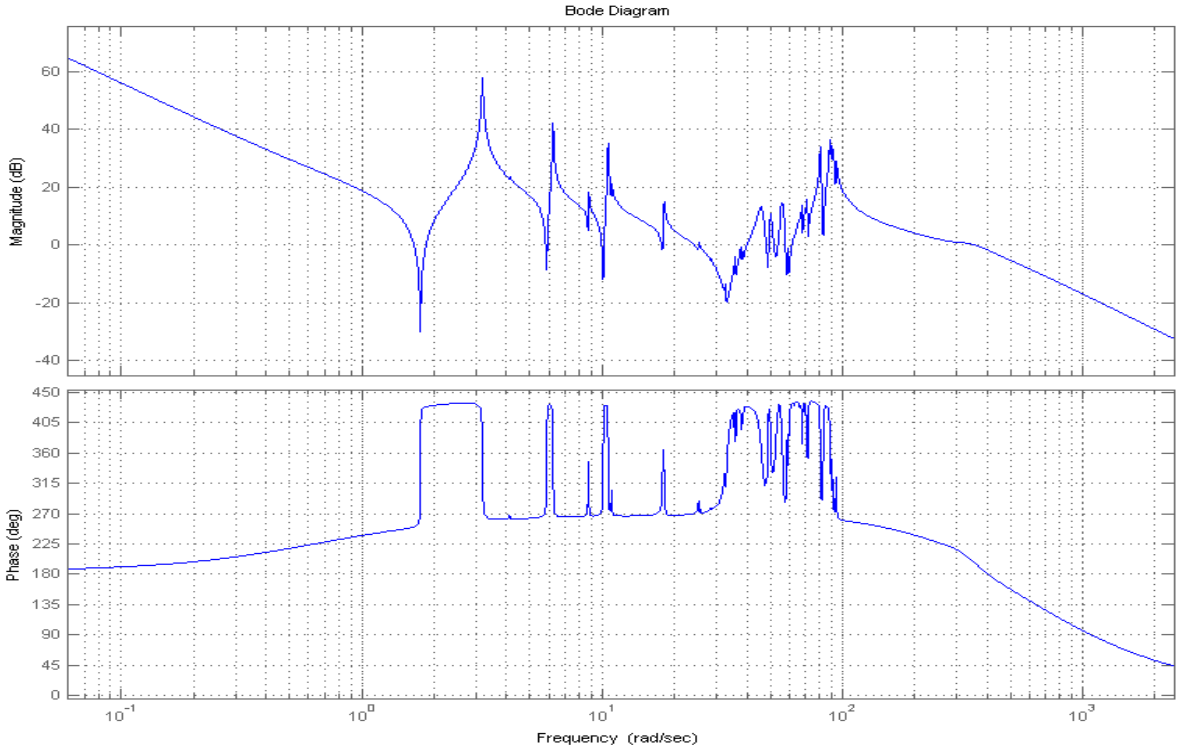


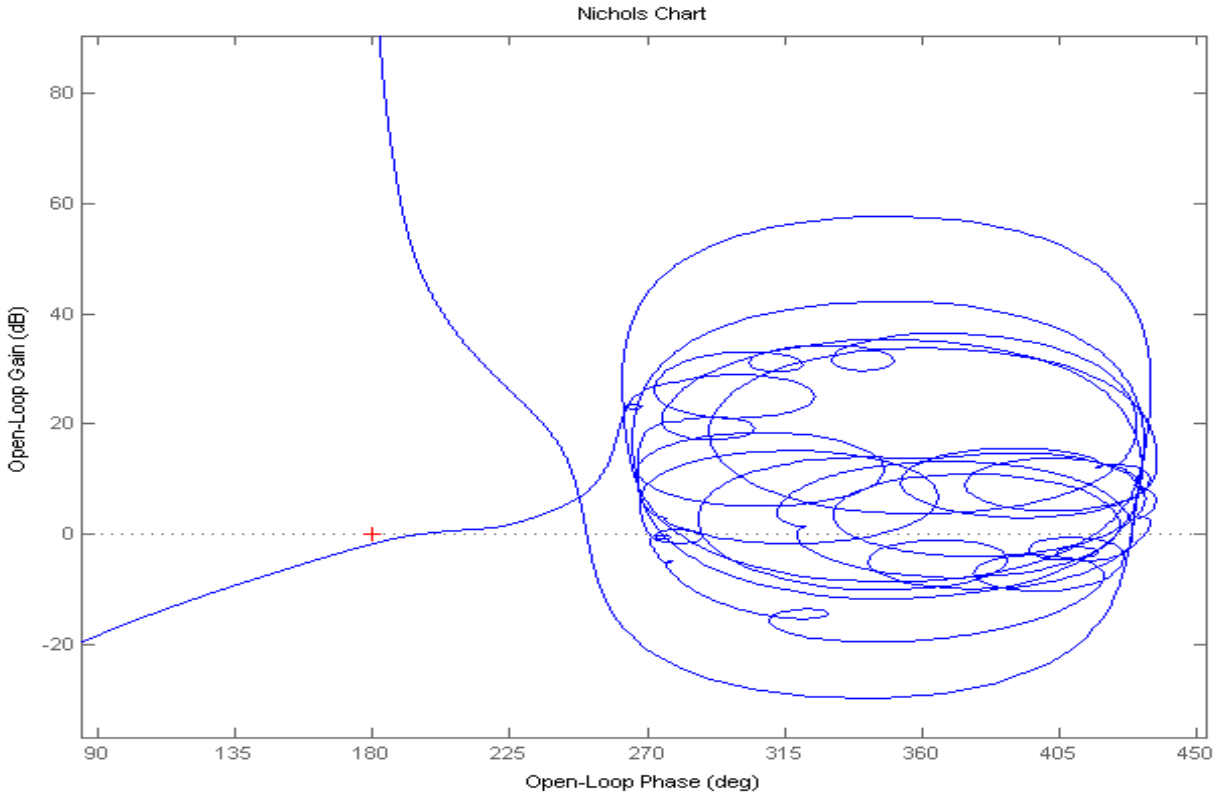
Figure 3.5.5 Simulink Model "Open_Loop.mdl" Used for Frequency Response Analysis

The frequency response and stability analysis results shown in Figure 3.5.6 are identical to the frequency response results obtained in the previous section in Figure 3.4.3.9. The stability margins shown are poor but they have been improved with filters in more recent simulation files.

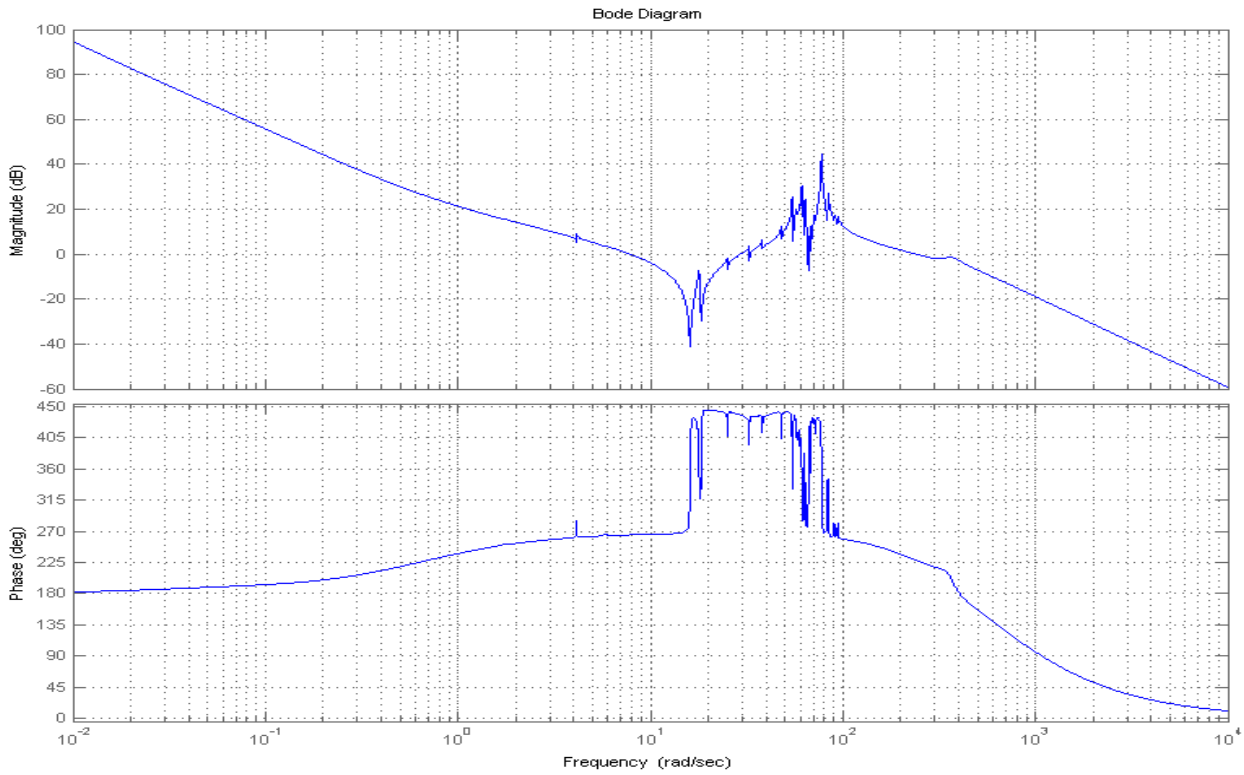
Roll Axis Open-Loop Frequency Response



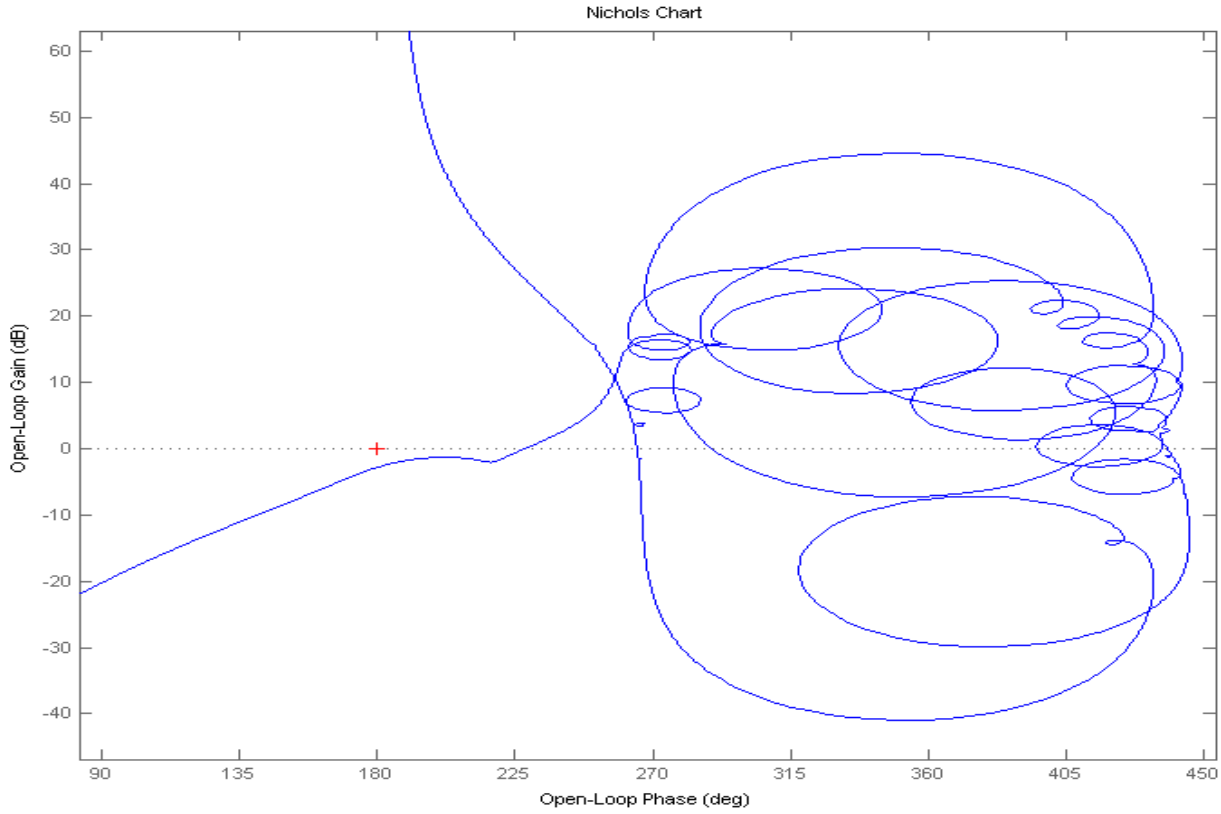
Roll Axis Stability



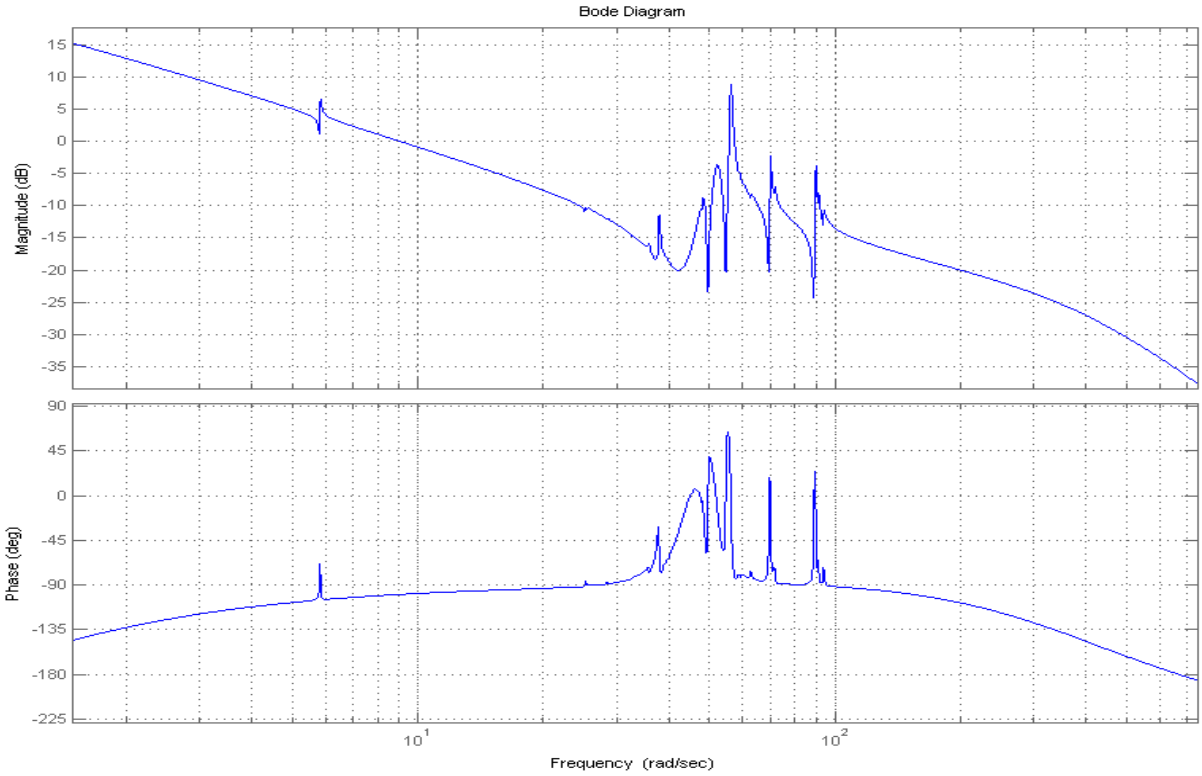
Pitch Axis Open-Loop Frequency Response



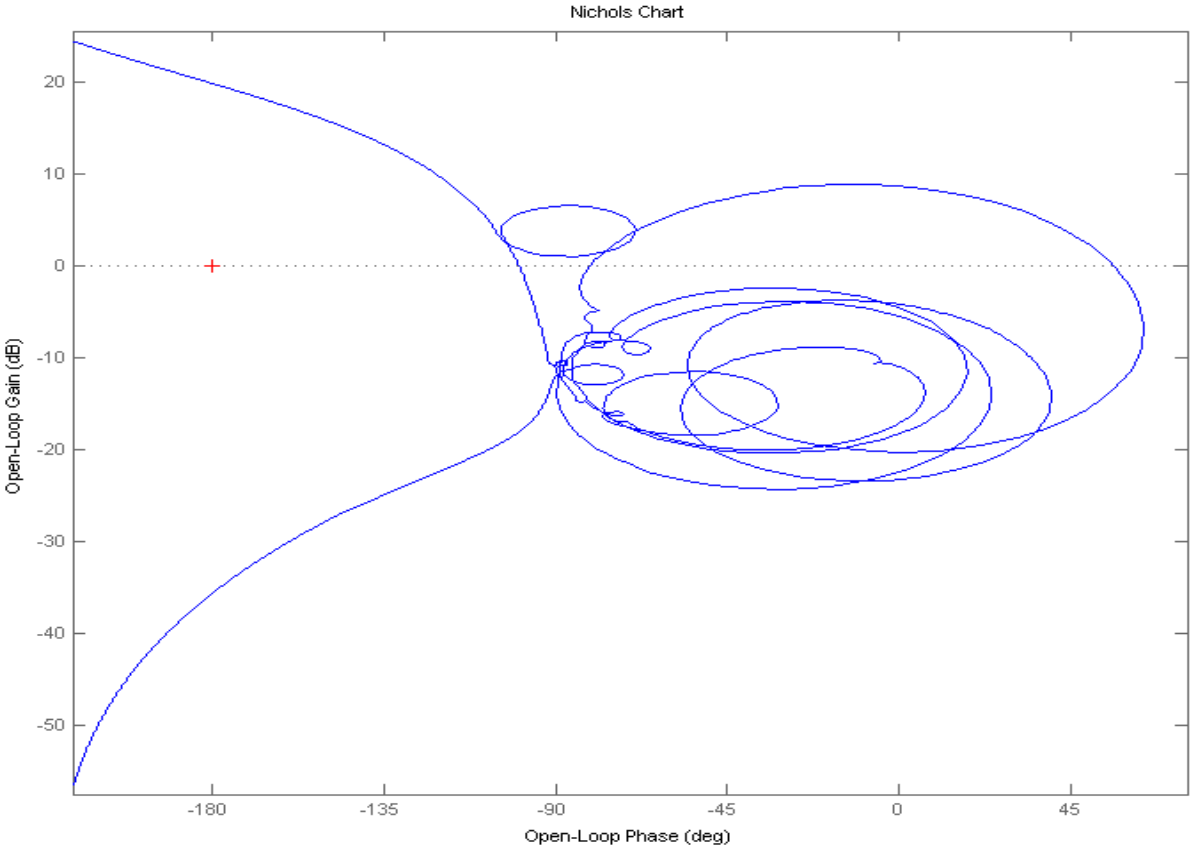
Pitch Axis Stability



Yaw Axis Open-Loop Frequency Response



Yaw Axis Stability



Robustness Analysis of the Four SG-CMG Spacecraft

Robustness is the ability of the system to tolerate uncertainties and variations, either internal or external. Our next step is to analyze the spacecraft control system's robustness to internal parameter variations. The question is, how much parameter variations can a system tolerate before it becomes unstable, or stops performing properly? Parameter uncertainties are imprecise knowledge of the plant model parameters, such as in this case, the mass properties, moments of inertia, CMG momentum, momentum direction, gimbal angle and direction, etc. The structured uncertainties in a model are specified in terms of variations in the actual plant parameters above and below their nominal values. We will use the IFL method to "pull" the uncertainties out of the plant M(s). The uncertainties are represented by a diagonal block Δ that is connected to the plant by means of some additional inputs and outputs, as shown in Figure (1). An input/ output pair for each parameter variation (δ_i).

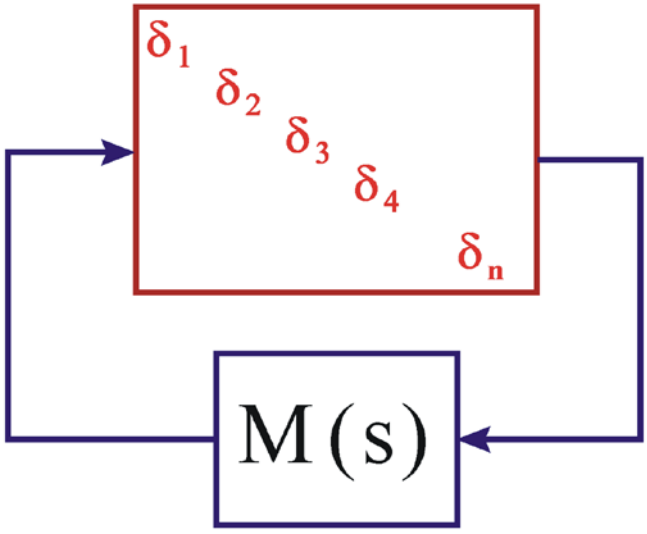


Figure 4 Closed-Loop Plant M(s) with the uncertainties block "pulled out"

The IFL methodology is implemented in Flixan. The program reads the nominal plant parameters for the spacecraft and the CMGs from an input file. It also reads the uncertainties for some of the parameters from the same file. For every non-zero parameter variation the program creates an additional input/ output pair that is supposed to hook up to a (δ_i) element in the block. The magnitude of each element represents the maximum possible variation of the parameter above or below its nominal value. In essence we create (n) additional inputs and outputs to the plant that connect to the uncertainties block Δ, which is a block diagonal matrix $\Delta = \text{diag}(\delta_1,\delta_2,\delta_3,...\delta_n)$. For convenience the diagonal block Δ is normalized so that its individual elements now vary between +1 and -1. We also normalize the plant M(s) by scaling its input/ output elements as needed to connect with the normalized Δ whose elements are now bounded to ±1. The individual elements of Δ may be scalars or matrices

and each represents a real uncertainty in the plant. Some parameter variations which are higher than rank-1 dependency may generate two or three (δ_i). An Ixz cross-product of inertia, for example, will couple in both, roll and yaw axis, creating two separate δ_i's, one in roll and another in yaw axis. In this case we treat them as two separate uncertainties. M(s) represents the known dynamics consisting of the plant model with the control system in closed-loop form. The augmented state-space system is used to perform robustness analysis using μ. The system in this configuration is defined to be robust if it remains stable despite all possible variations in the Δ block as long as the magnitude of each individual variation is bounded below $\pm\delta_{(i)}$, or ±1 in the normalized system. The structured singular value (μ) is the perfect tool for analyzing this type of robustness problems in the frequency domain. The value of 1/μ(M) represents the magnitude of the smallest perturbation that will destabilize the normalized system M(s).

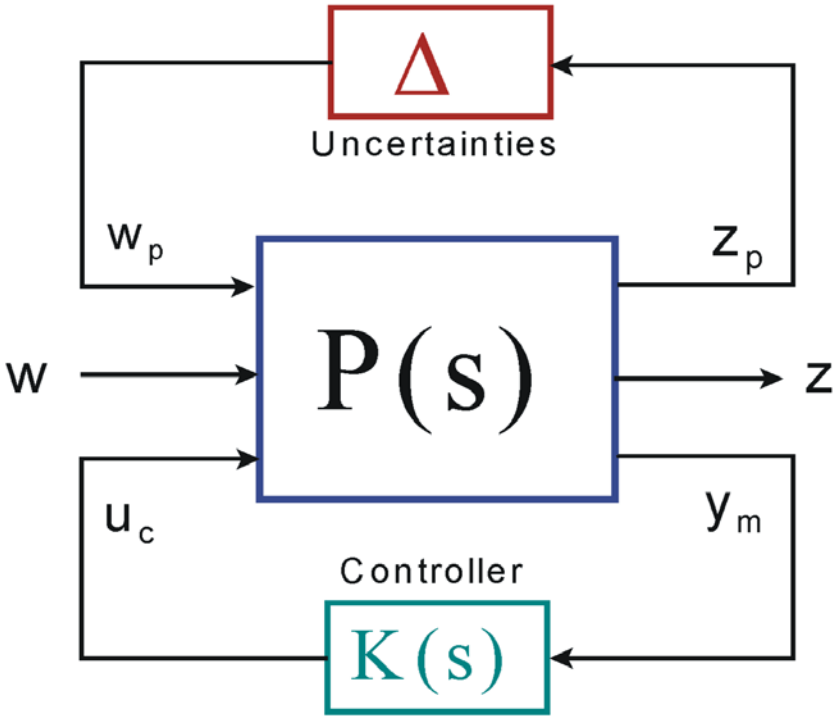


Figure1a Robustness Analysis Model

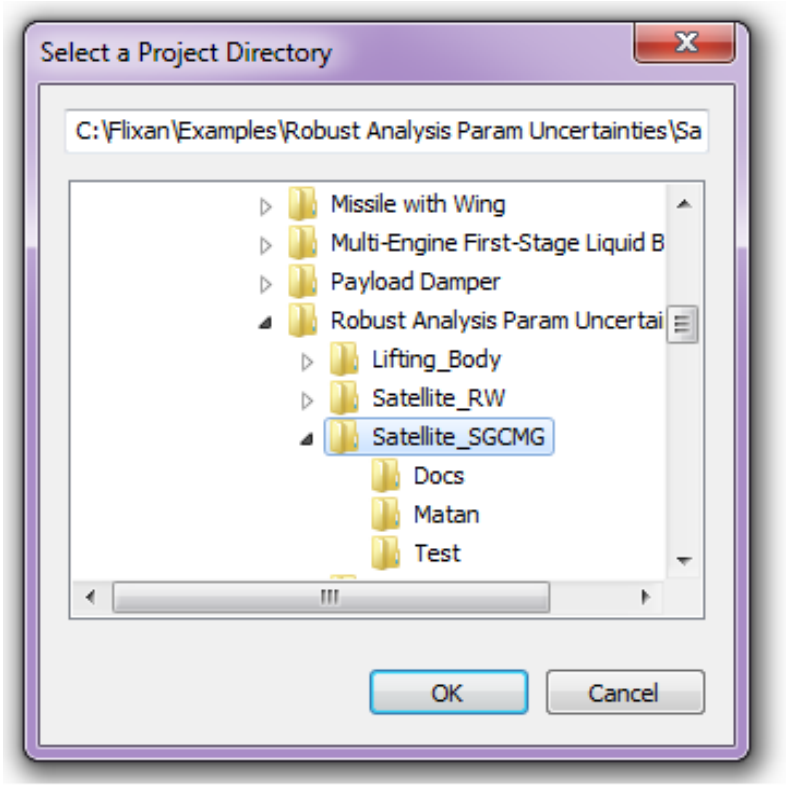
The block diagram formulation, shown in Figure (1a), is used for μ-synthesis or robustness/ performance analysis using (μ) methods. The plant has inputs and outputs that connect to the uncertainty block. It also has inputs and outputs that connect to the control system K(s). It also has external disturbances (w) and output performance criteria (z) which are also normalized to unity. We say that the plant meets sensitivity requirements when the mu frequency response between (w) and (z) is less than one at all frequencies. Similarly, and according to the small gain theorem, the closed-loop system is robust to the specified uncertainties as long as μ(M) across the normalized block Δ, (that is, between w_p and z_p) is

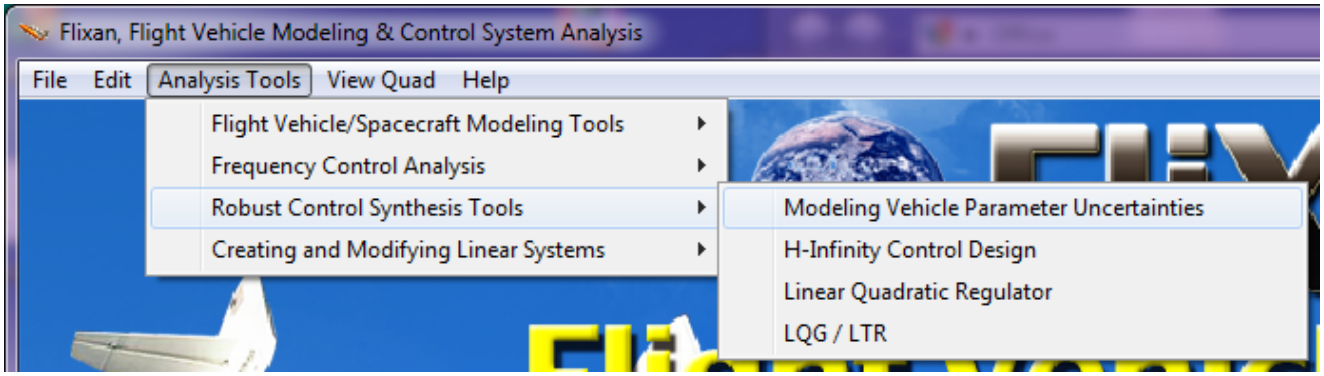
less than one at all frequencies. Robust performance is when it satisfies both simultaneously, that is, the mu frequency response between [wp, w] and [zp, z] is less than one at all frequencies.

The analysis files for this example are in folder: "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(g) 4SGCMG Robust_Anal*". The Flight Vehicle Modeling program (FVP) creates two spacecraft models, a nominal model, and a similar one that has 61 additional inputs and outputs for the 61 parameter uncertainties. Dynamically both models are the same, except for the second model includes 61 internal parameter uncertainties using fictitious inputs and outputs. We use the nominal model to do stability analysis and to prove that the nominal system is stable and then we use the model with the uncertainties to do robustness analysis. We must close the control loop (without commands) and, as long as the system is stable, perform μ-analysis across the uncertainty inputs and outputs to check if there is any combination of uncertainties that will drive it unstable. The Matlab analysis is performed in folder " *CMG Control\(g) 4SGCMG Robust_Anal\Matan*". The spacecraft parameters are in the input file "*FlexSc_4CMG.Inp*". The file includes also a set of modal data consisting of 40 selected modes. The mode selection process is not shown here because it is fully described in another example. The input file also includes the parameter uncertainties in a separate set of data, title: "*Uncertainties for Flexible Agile Spacecraft with 4 SG-CMG*". The parameter uncertainties are additive variations to the nominal spacecraft parameters. Notice that, not all parameters should be varied and the user must use caution in selecting which parameters to vary, because perturbing some parameters does not create a plant variation, and this causes an error in the program. In this example we vary the spacecraft moments and products of inertia, we vary the frequencies in 8 flex modes, that is, the strongest 8 modes. We also add uncertainty in the CMG momentum ±50 (ft-lb-sec), the momentum direction, the initial gimbal angle (δ_0), the CMG moment of inertia about the gimbal axis (J_g), and the pyramid surface orientation angles (β, γ). We did not include variations in the gimbal directions and the CMG inertias (J_s, J_o) because they do not create plant variations, that is, only in this example.

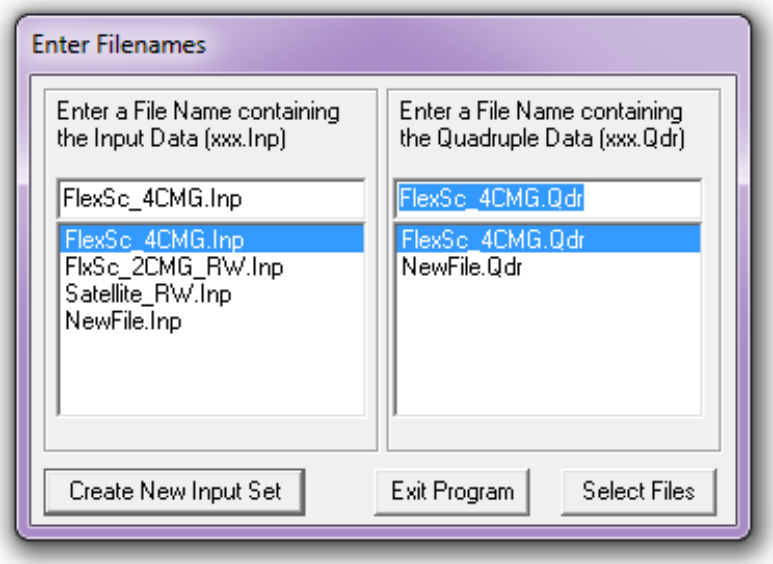
Spacecraft Model Creation

To generate the spacecraft models we will skip the input data file creation process and just use the existing spacecraft and uncertainties data which are already saved in the input file "FlexSc-4CMG.Inp". We start the Flixan program and select the current project folder: "*\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(g) 4SGCMG Robust_Anal*". Then from the main menu select "*Analysis Tools*", "*Robust Control Synthesis Tools*", and "*Modeling Vehicle Parameter Uncertainties*", as shown below.

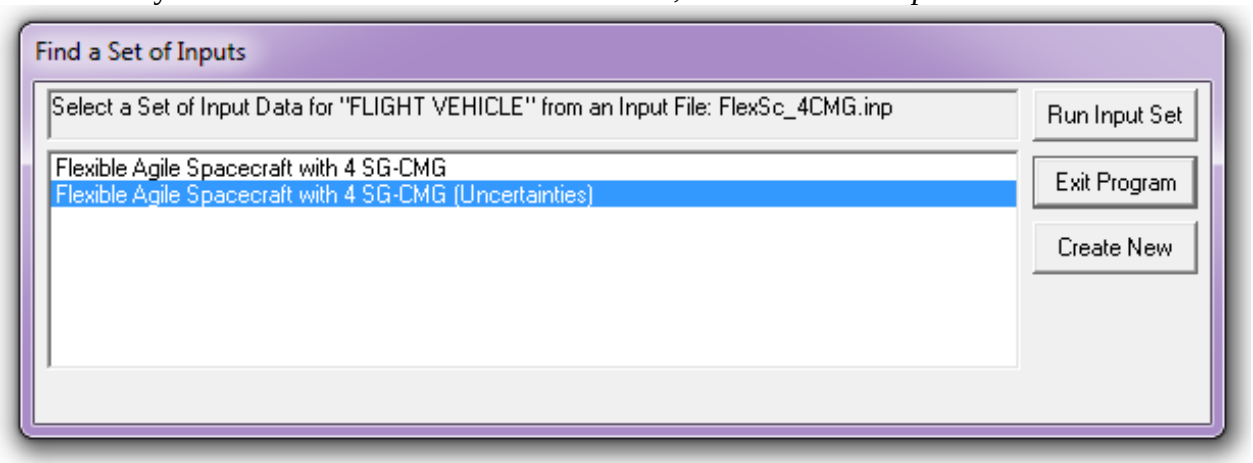


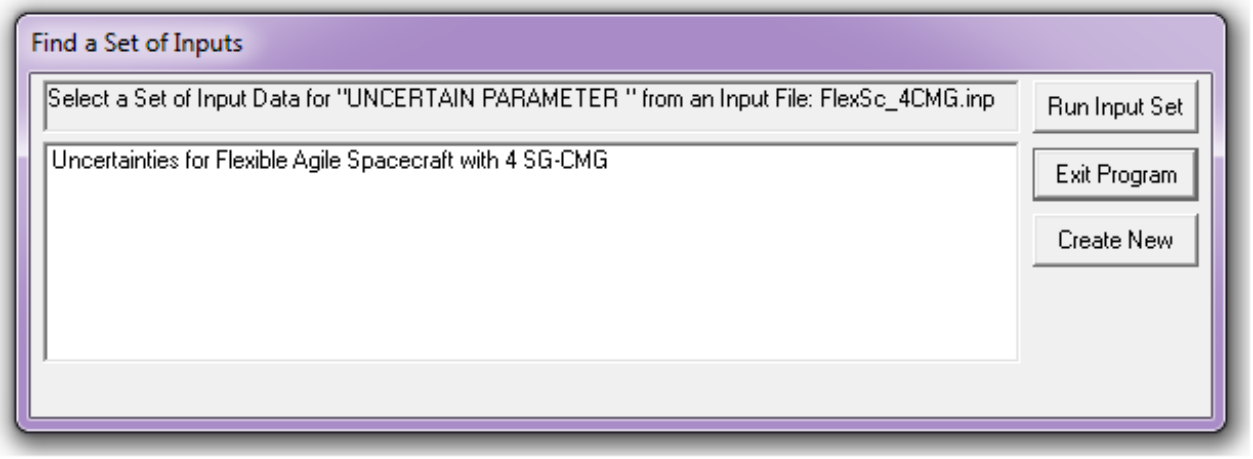


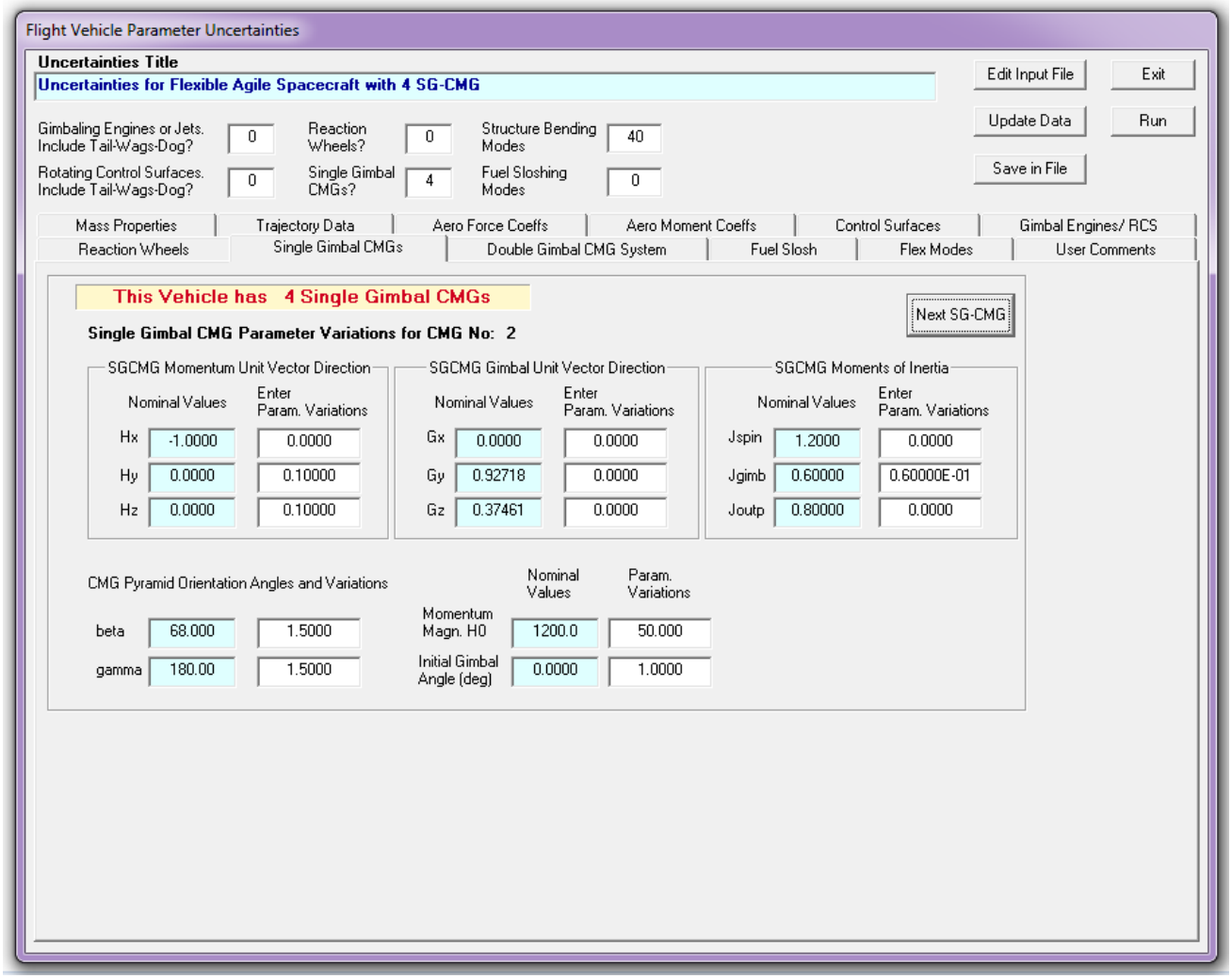
From the filename selection menu select the input data file "*FlexSc_4CMG.Inp*", and the output systems file "*FlexSc_4CMG.Qdr*", and click on "Select Files" button.



In the following menu choose the title of the data set that contains the spacecraft and CMG input data in the nominal configuration and from the following menu chose the uncertainties data set. There is only one set of uncertainties to choose from, and click "*Run Input Set*" in both cases.



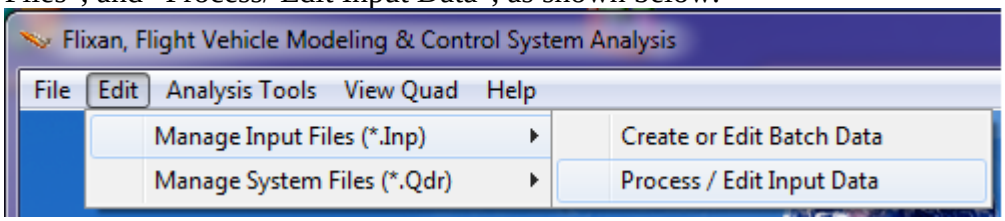




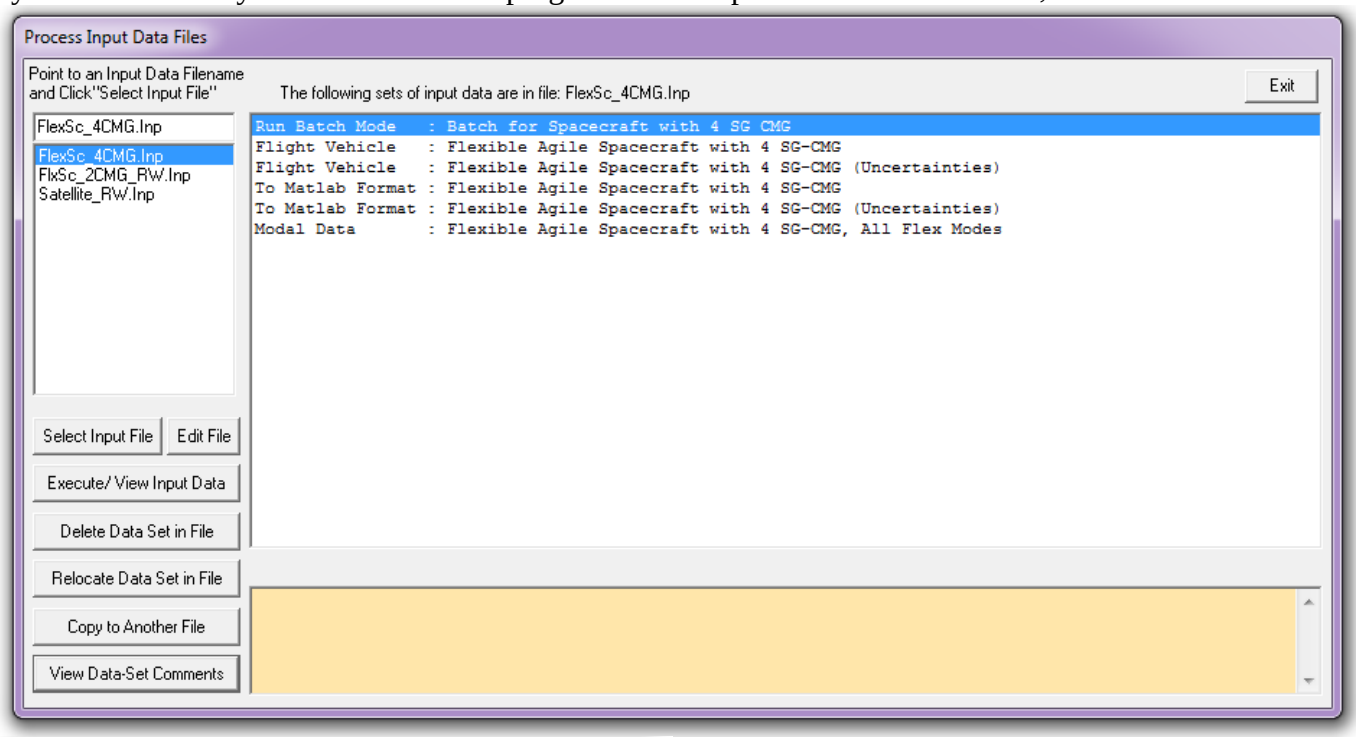
The program displays the above dialog that consists of various tabs where the user may check the validity of vehicle and uncertainties data. Each tab consists of a group of data. In the tab selection above, the 4 Single-Gimbal CMG data and variations are shown. The parameters are in the light blue fields on the left and the parameter variations are on the right. You may also check the pass properties data, and the flex mode variations by clicking on the corresponding tabs.

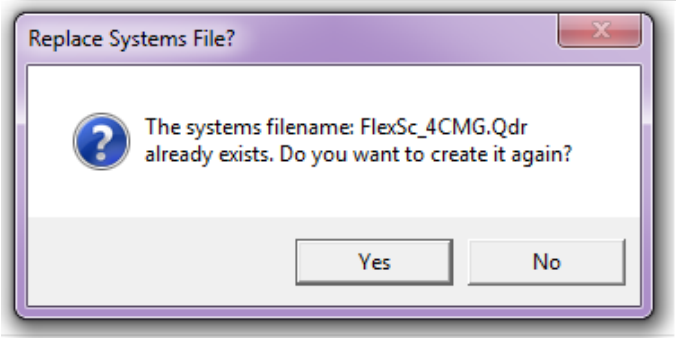
Click on "Run" and the program generates the agile satellite state-space model and it will save it in file "*FlexSc_4CMG.Qdr*". It is also exported in Matlab format as a state-space function "*sc_4cmg_flex_unc.m*" for μ-analysis in Matlab. This system includes the 61 additional inputs and the 61 additional outputs that connect to the uncertainty block Δ. In our μ-analysis we don't connect it with a Δ block, but we simply compute the μ frequency response between the 61 inputs and outputs. Notice, that the input/ output pairs connecting to the uncertainty block Δ is greater than the number of parameter variations. This is because some of the uncertainties are higher than rank-1 dependency, and they couple into more than one direction since they appear in 2 equations. The products of inertia, for example, couple in two directions. Since we do not decouple the system but analyze roll, pitch, and yaw axes together, it is acceptable to treat then as two separate parameter variations although they originate from a single parameter variation.

A much faster way to generate the spacecraft models is to run the batch set "*Batch for Spacecraft with 4 SG CMG*" located on the top of the input data file. Start the Flixan program, and select the project folder "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(g) 4SGCMG Robust_Anal*" as before. Then, from the Flixan main menu, go to "Edit", "Manage Input Files", and "Process/ Edit Input Data", as shown below.



The program displays the following Menu/ Dialog. From the file selection menu on the left select the input file "*FlexSc_4CMG.Inp*". The menu on the right shows the titles of the data-sets which are included in the input file. Select the batch on the top "*Batch for Spacecraft with 4 SG CMG*", and click on the "*Execute Input Data*" button to process the batch set. The Flixan program will generate the systems and save them in systems file "*FlexSc_4CMG.Qdr*". If a previous version of the systems file already exists in folder the program will ask permission to recreate it, answer "Yes".





The batch performs the following operations. It creates a nominal spacecraft model with 40 flex modes, title: "*Flexible Agile Spacecraft with 4 SG-CMG*" in file "*FlexSc_4CMG.Qdr*". It also creates the perturbation model with the 61 uncertainties, title: "*Flexible Agile Spacecraft with 4 SG-CMG (Uncertainties)"*. The two state-space models are then converted to Matlab system functions, "*sc_4cmg_flex.m*" and "*sc_4cmg_flex_unc.m*", respectively that can be loaded into Matlab. Both systems contain roll, pitch, and yaw coupled vehicle dynamics. The second system, in addition to the standard inputs and outputs of the first system, it includes also the 61 pairs of inputs and outputs that connect to the uncertainties Δ block. This is the diagonal block that contains the normalized uncertainties which vary between -1 and +1. Some of the uncertainties couple only in one axis, but some uncertainties couple in more than one axis.

Simulation Model

Now, let us take a look at a linear simulation model that uses the nominal spacecraft dynamics without parameter variations. This simulation model is in file "*Lin_Flex_Sim.mdl*", shown in Figure (2). The model is initialized using file "start.m", which also loads the two spacecraft systems into Matlab workspace.

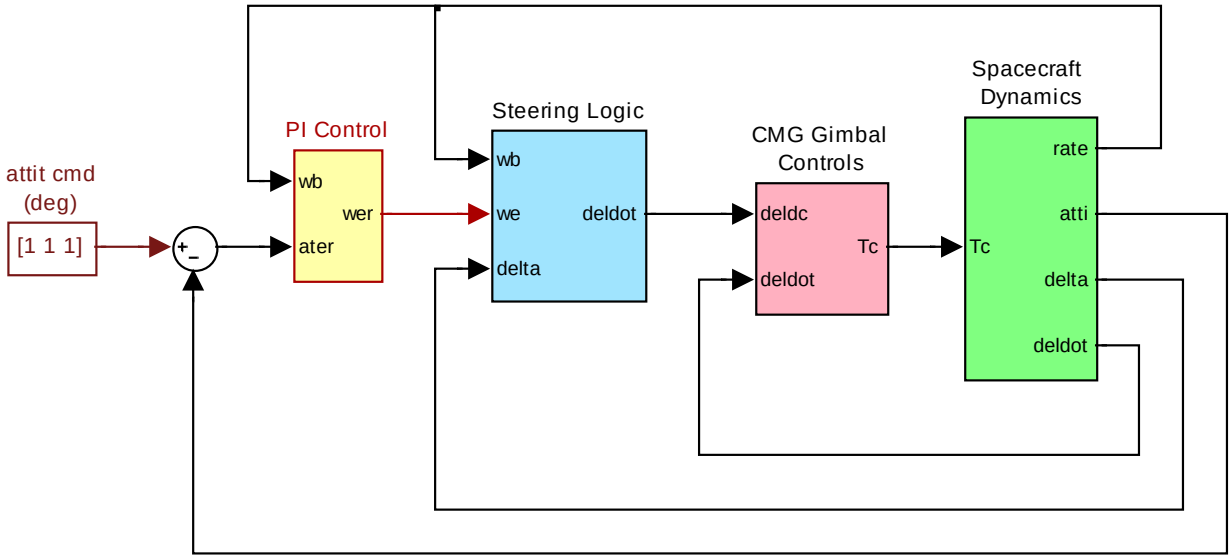


Figure 2 Simulink model for the Flex Agile Spacecraft with 4 SG-CMGs, in file "*Lin_Flex_Sim.mdl*"

The spacecraft dynamics (green) block consists of the nominal state-space system "*sc_4cmg-flex.m*" (no uncertainties). The input is a vector of four CMG gimbal torques which control the gimbal rates. The outputs are: spacecraft attitude, rates, CMG gimbal angles, and gimbal rates. The gimbal rate commands come from the CMG steering logic, and the gimbal rate control system provides the gimbal torques required to control the rates. The purpose of the steering logic is to control the spacecraft rate by creating gimbal rate commands at the 4 CMG gimbals. The inputs to the steering logic are: spacecraft rates, gimbal angles, and spacecraft rate error. The attitude control system is a simple PI. The (D) part of the PID is included in the steering. In the simulation the spacecraft is commanded to perform a one degree rotation in all 3 directions. The purpose of the simulation is to demonstrate nominal system stability.

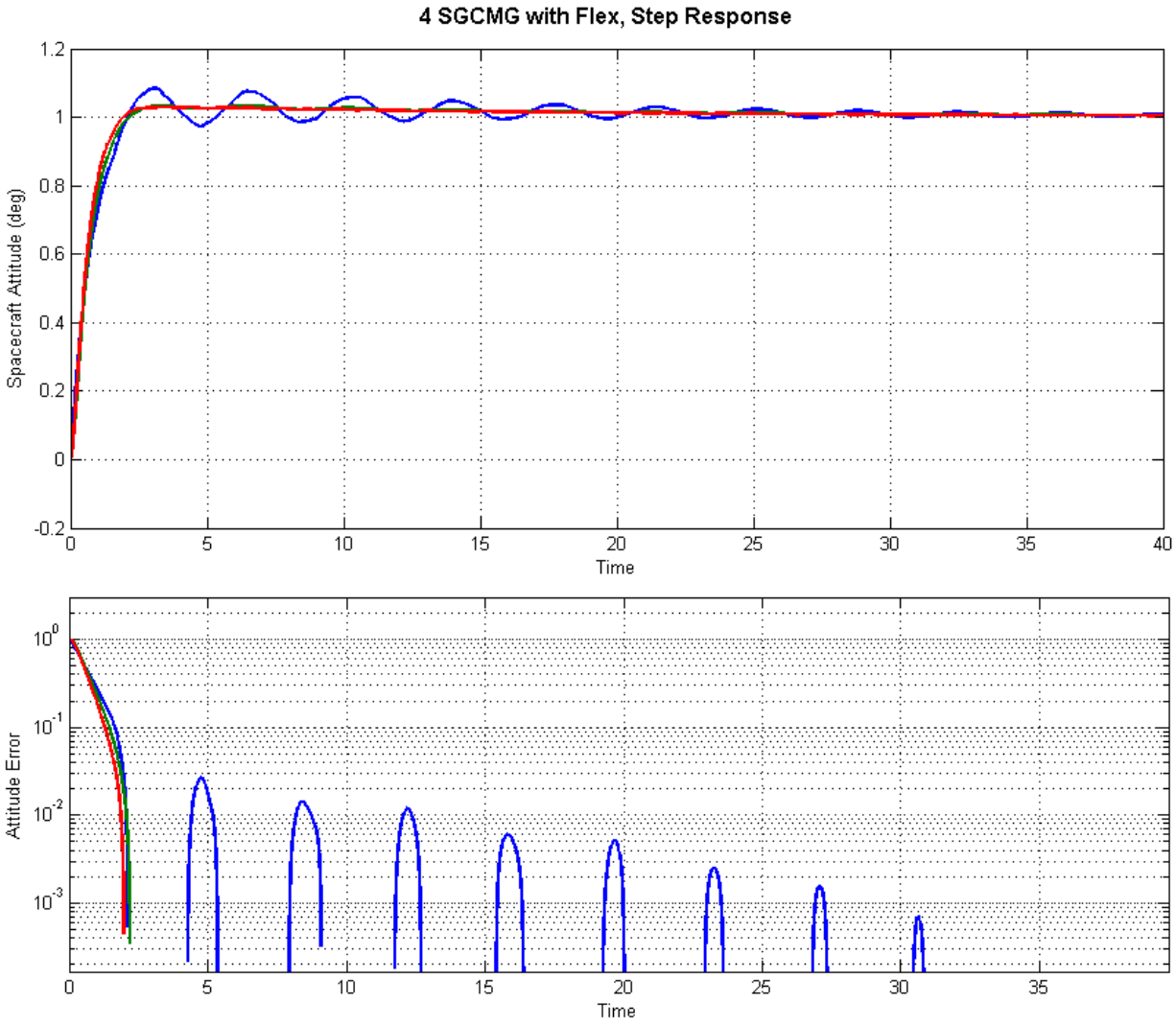


Figure 3a Spacecraft attitude response to one degree command in all 3 directions

Figure (3a) shows a stable attitude response to 1º command in all 3 directions. The roll axis (blue) takes longer to settle because flexibility is stronger in roll. Figure (3b) shows the spacecraft rate and acceleration at two separate locations with different flex mode sensitivity. Figure (3c) shows the gimbal angles and gimbal rates. It also shows the CMG momentum. The roll momentum oscillates as the CMGs respond to the roll structural oscillations.

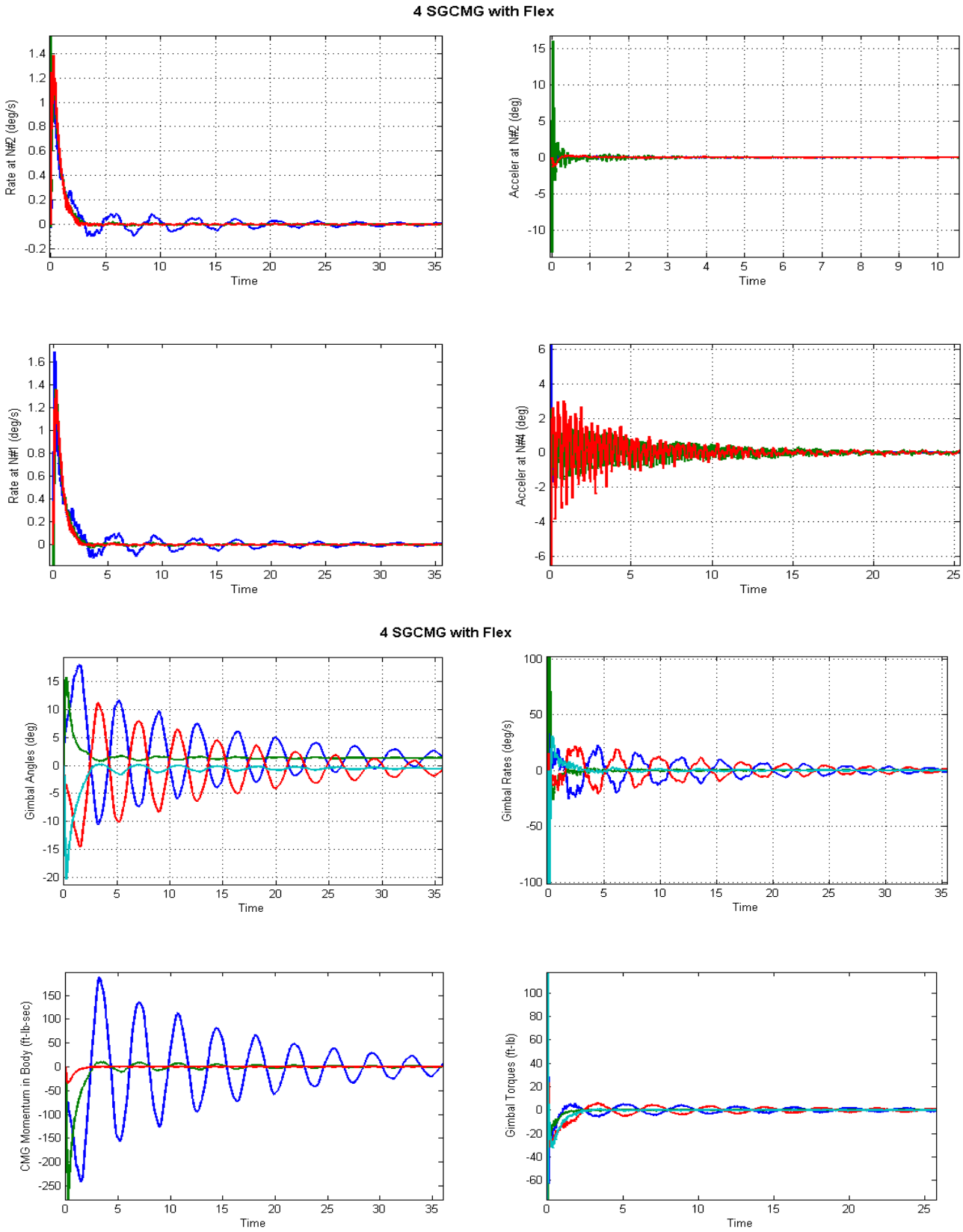


Figure (3b & 3c) Spacecraft response to the one degree command in 3 directions

Linear Stability Analysis

We also perform linear stability analysis to determine the phase and gain margins of the nominal configuration. The file "freq.m" performs the open-loop frequency response analysis using the Simulink model "*Open_Loop.mdl*", shown in Figure (4). This model uses the same subsystems as the closed-loop simulation model, but has one axis loop opened and the other two closed. The Matlab script linearizes (using the linmod function) the system across the opened input and output, computes its frequency response and plots the Nichols plots, as shown in Figures (5a-5c).

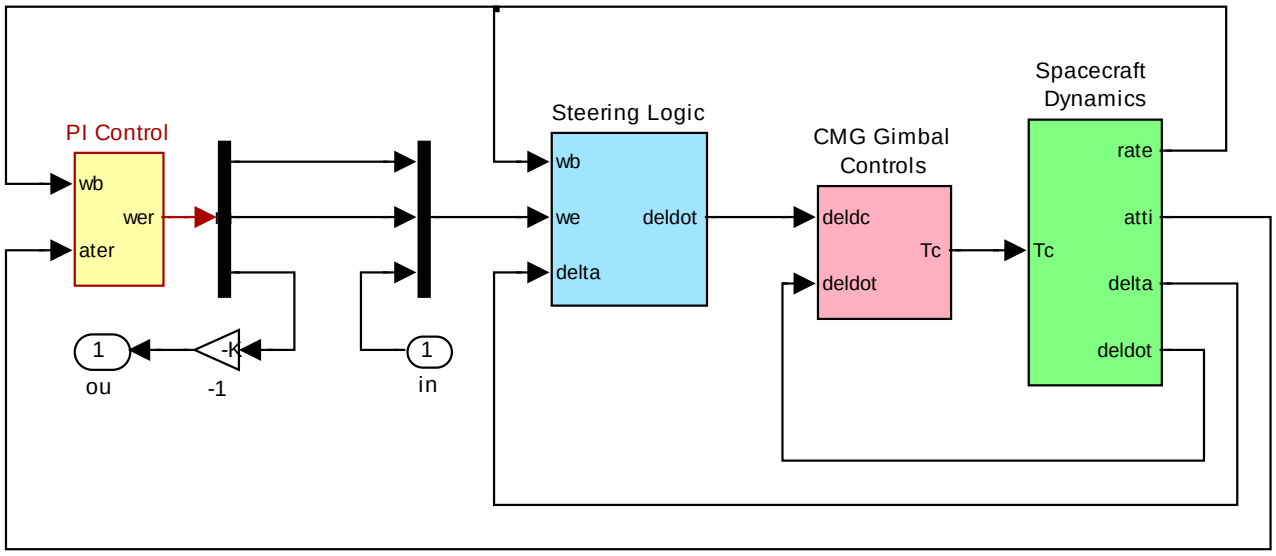
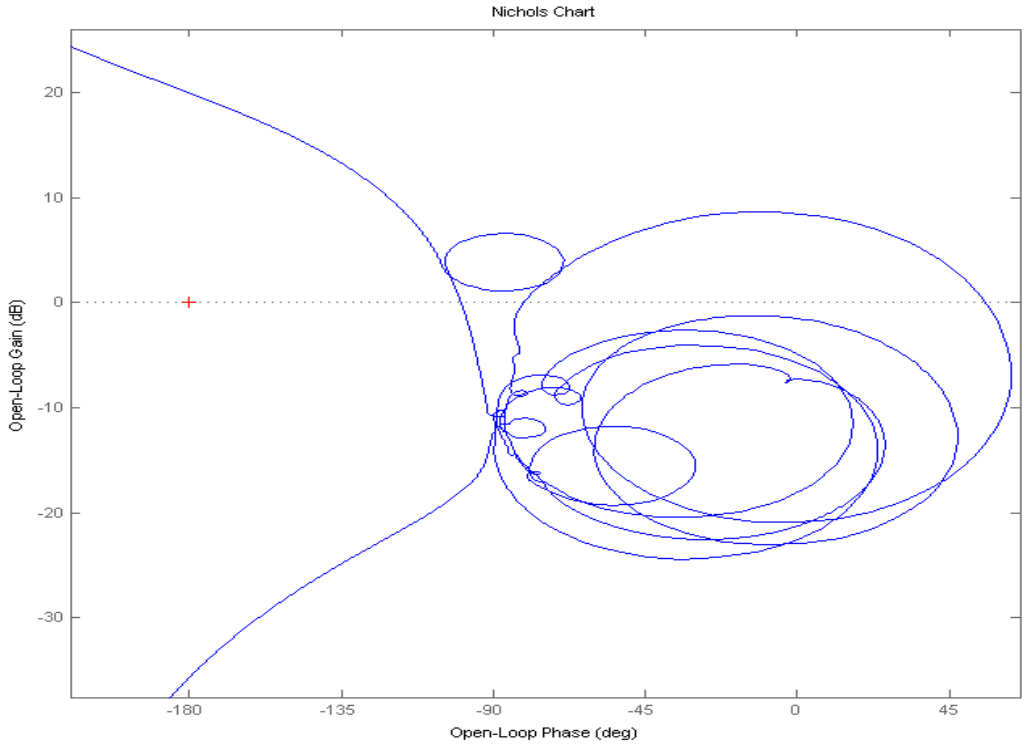
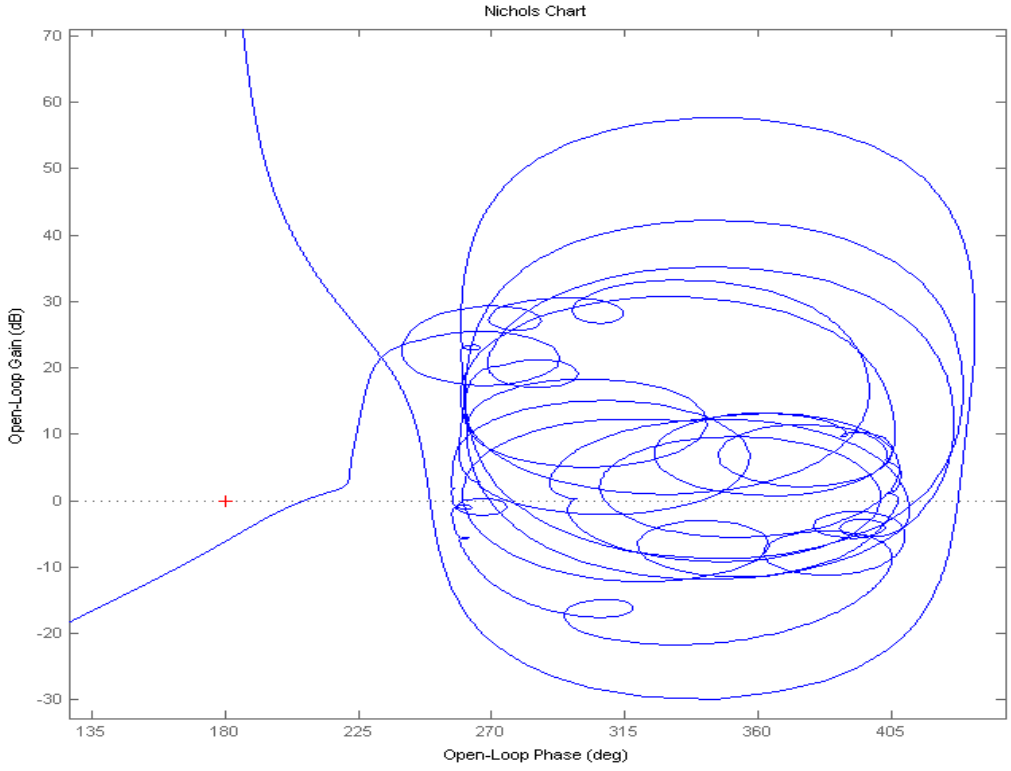


Figure 4 Simulink model "*Open_Loop.mdl*" used for linear stability analysis

Nominal System Stability Margins in the Yaw Axis



Nominal System Stability Margins in the Roll Axis



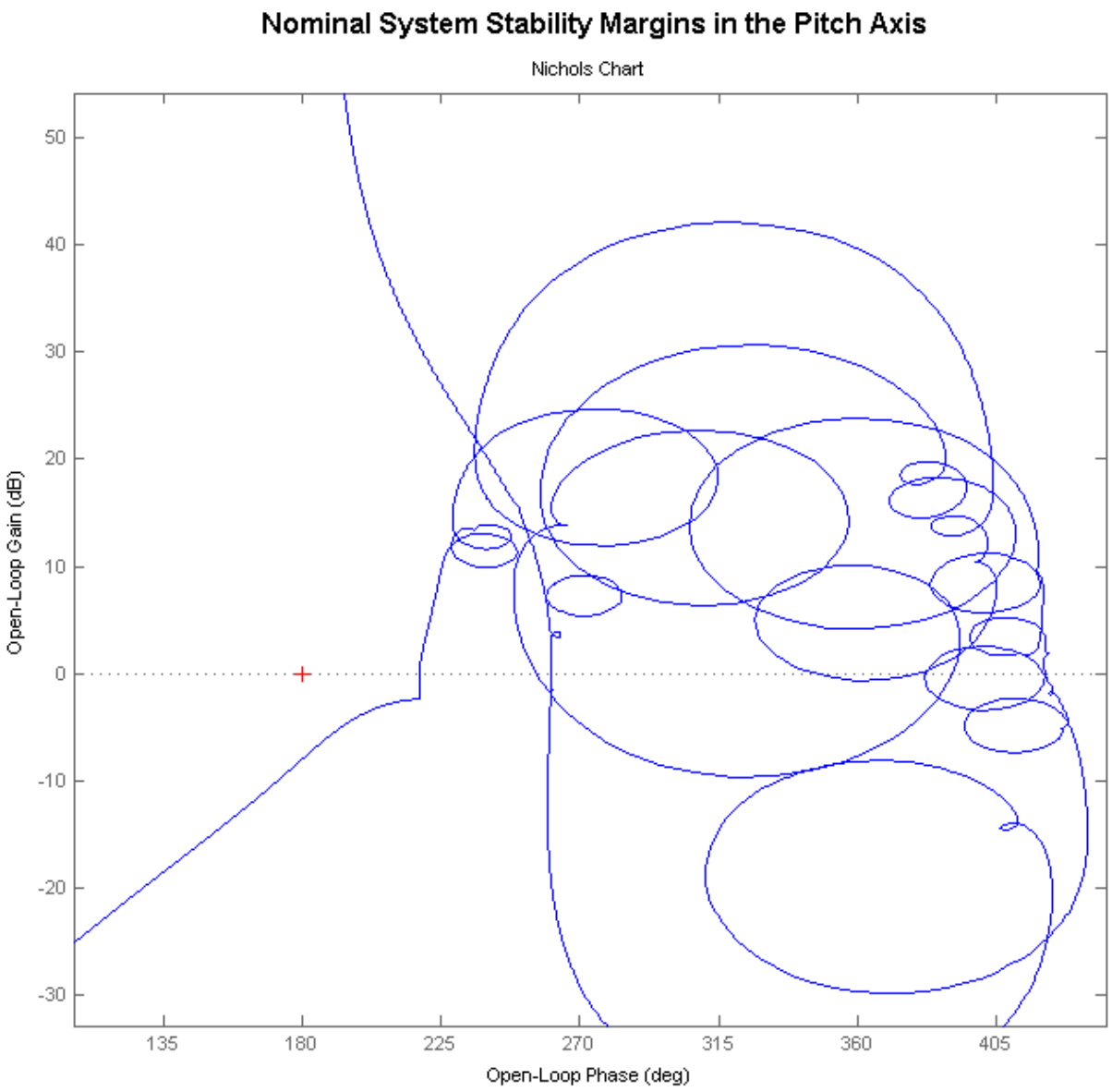


Figure 5c System is nominally stable in all 3 axes

Robustness Analysis

Robustness analysis is performed by closing the attitude control loops (assuming that the closed-loop system is stable) and by calculating the μ-frequency response of the system across the normalized perturbations block Δ. Since the diagonal elements of Δ do not vary more than ±1, the system is assumed to be robust when the SSV of the closed-loop system across the diagonal perturbations is less than one at all frequencies. The following Simulink model "Robust-Anal.mdl" is used to calculate the mu.

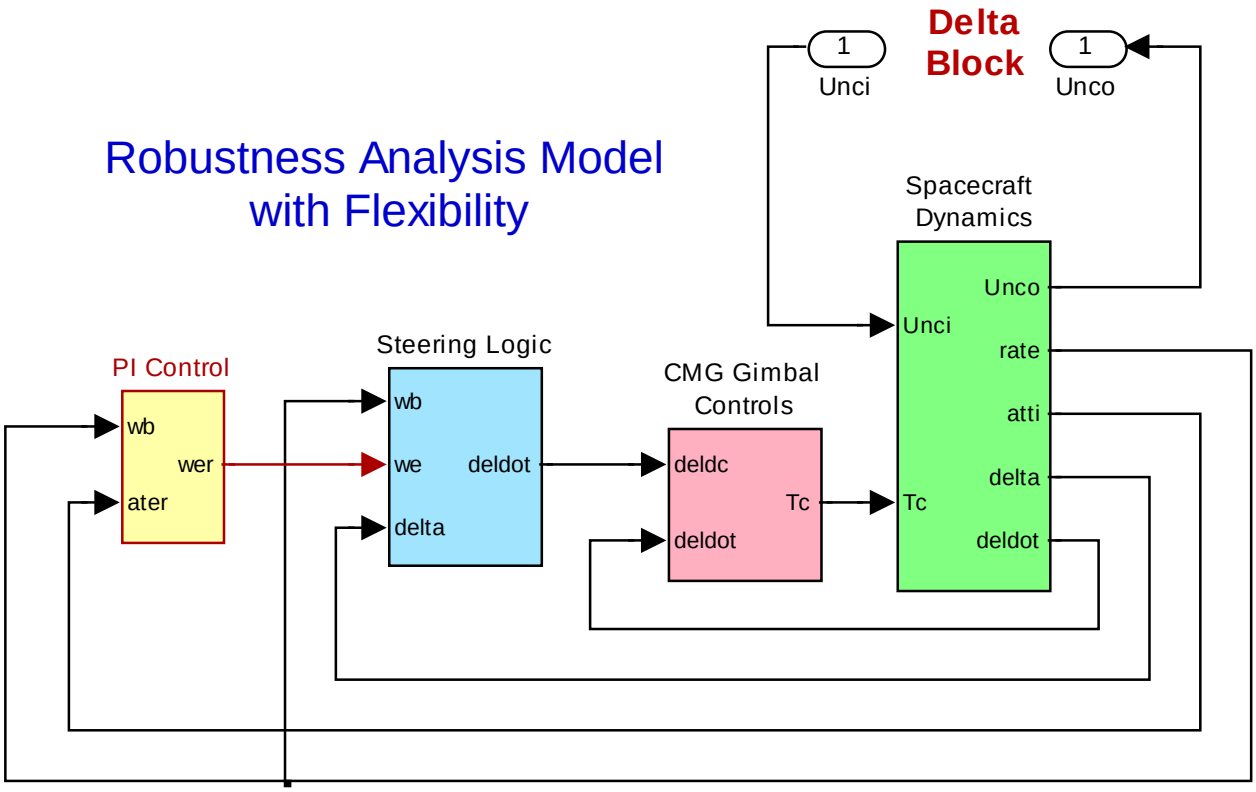


Figure 6 Simulink model "Robust_Anal.mdl" used to calculate system robustness to parameter variations

All subsystems are the same as the ones used in previous models, except for the spacecraft block which now includes the additional inputs and outputs that hook up to the uncertainty block. The spacecraft subsystem is shown in detail in Figure (7). It uses the state-space system "*sc_4cmg_flex_unc.m*" which includes the 61 additional inputs and outputs which model the uncertainties. The Matlab script "freq.m" calculates also the μ-frequency response of the linearized system in Figure (6) assuming that the parameter variations are "real" (not "complex" because complex variations are very conservative specially on flex mode variations). Figure (8) shows the μ-frequency response across the Δ block. It is less than one at all frequencies concluding, therefore, that the system is robust to the parameter variations.

Figure 7 Spacecraft System from "*sc_4cmg_flex_unc.m*" that includes the uncertainty inputs and outputs.

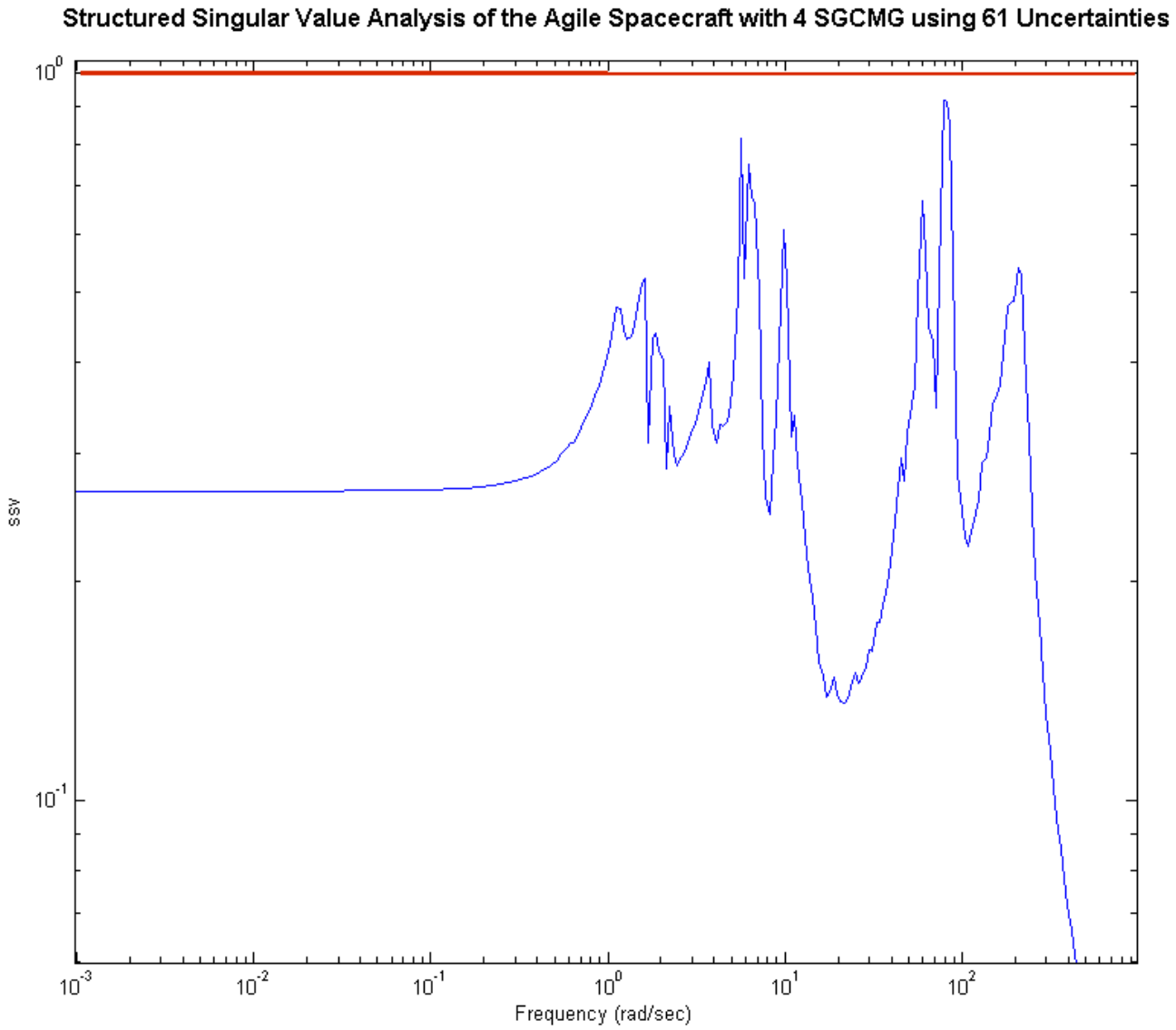


Figure 8 The Structured Singular Value frequency response across the perturbations block Δ is less than one at all frequencies, therefore, system is robust to parameter variations.

3.6 Spacecraft Configuration Using 2 SG-CMG and a RW

In the previous section we analyzed spacecraft configurations using four SGCMG that perform 3-axis attitude control of the spacecraft for quick maneuvering between targets. The control law is efficient, symmetrical and fast because it uses the max control torque and momentum capability of the CMG devices as it performs eigenaxis maneuvers, assuming that the pointing requirement is the same in all directions. But what if the maneuvering requirements on the spacecraft are not the same in all directions? For example, if we want to point an antenna or a beam of light (which is along the spacecraft x axis) in a certain direction we are more interested in the pitch and yaw efficiency of response and much less in roll, since roll errors do not affect antenna operation. In addition, the spacecraft which is normally pointing nadir (towards the earth center) is required to maneuver not more than 40 degrees from nadir. Furthermore, the spacecraft moment of inertia in roll is much less than in pitch and yaw axes, so the torque and momentum requirement in roll would be significantly less than pitch and yaw. It is conceivable, therefore, and since the SGCMG are very costly devices that we may be able to get by with fewer momentum control devices.

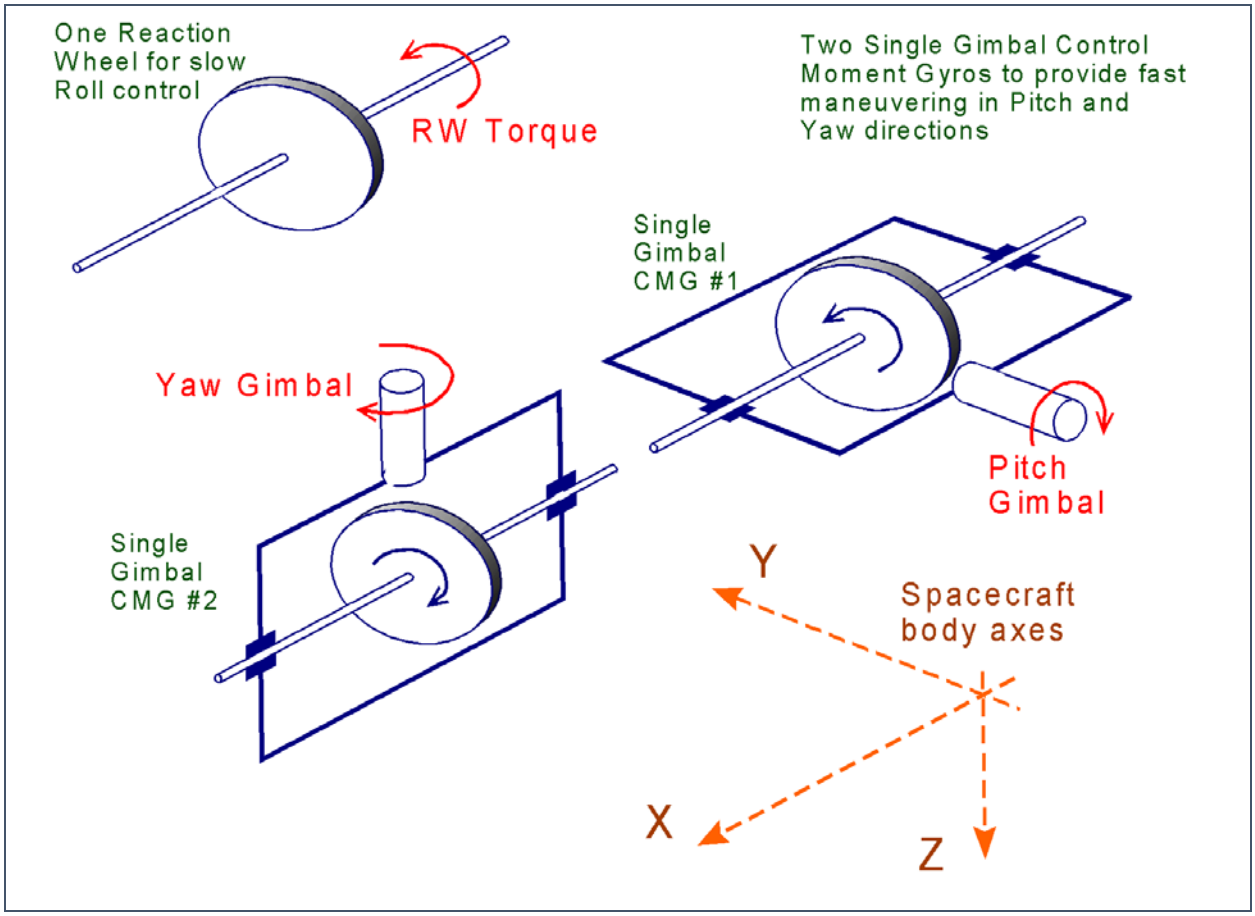


Figure 3.6.1 The Momentum Control System in this configuration consists of one Reaction Wheel combined with two Single-Gimbal CMGs

The Momentum Control System (MCS) described in this configuration consists of two single-gimbal CMGs for pitch and yaw and only one reaction wheel for roll control, see Figure (3.6.1). The RW momentum (spin) direction is parallel to the spacecraft roll axis and controls roll attitude. The two SG-CMGs are spinning at constant rate and their initial (nominal) momentum directions are along the +x and –x axis respectively, cancelling each other's momentum. Between operations there are occasional momentum dumps using the RCS jets to prevent the MCS momentum from reaching saturation. The momentum desaturation system is attempting to keep the CMG gimbals in the nominal position and the RW speed at zero. Each CMG gimbals only in one direction relative to the spacecraft and their momentum directions vary as they gimbal. CMG #1 is gimbaling in pitch, and initially (when the gimbal is at zero and the momentum is along x) it generates a yaw torque. The second CMG is gimbaling in yaw and it generates a pitch torque when the gimbal is in its nominal zero position. When the CMGs are gimbaling at bigger angles, rolling torques are also generated which are counteracted by the reaction wheel control system. The torque output direction of each CMG varies. It is orthogonal to the momentum vector and the gimbal direction, and since the momentum direction is constantly changing the steering algorithm must keep track of the gimbal angles (δ_i) and calculate the gimbal rate commands for the two SG-CMGs. It calculates also the roll torque command for the RW.

The Line-of-Sight (LOS) direction of the antenna is along the spacecraft x-axis, and the main priority of the ACS is to point the LOS at the target as fast as possible, which requires pitch and yaw maneuvering. Positioning the spacecraft in roll is a secondary priority and it is acceptable if it takes longer to converge in roll in comparison to pitch and yaw. Attitude maneuvering in pitch and yaw should, therefore, be performed faster using the CMG’s, while the roll axis is controlled by the reaction wheel system which is slower. The CMG torque capability in pitch and yaw and the control system bandwidth is much greater than the RW torque and bandwidth controlling the roll axis and the MCS, therefore, cannot perform ideal eigenaxis maneuvers as it was demonstrated by the 4 SG-CMG control system in previous sections. For comparison purposes, the LOS pointing error will be shown separately from the roll error.

Spacecraft Dynamics

The spacecraft rotational acceleration $\dot{\omega}$ is a function of the internal reaction wheel torque (T_{RW}), the CMG internal torque ($\underline{T}_{CMG}$), and also the external torques (T_{ext})

$$J_{sc}\,\underline{\dot{\omega}} = \underline{T}_{CMG} + \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T T_{RW} - \underline{\omega} \times \left(J_{sc}\,\underline{\omega}\right) + \underline{T}_{ext}$$

Where: $\underline{T}_{CMG}$ is the CMG torque and T_{RW} is the reaction wheel torque in spacecraft body. The CMG torque is a function of the gimbal rates and angles as defined in equations (2.2 and 2.4). It consists of torques transmitted through the gimbals and gyroscopic torques transmitted through the bearings. The RW motor speed control dynamics is ignored and we assume that the RW torque is equal to the commanded torque from RW steering, but it is limited to less than ±10 (ft-lb).

The reaction wheel rate of change of momentum is a function of the reaction wheel torque which is in the x direction. I_W is the wheel moment of inertia about the spin axis.

$$\underline{\dot{H}}_{RW} + \omega \times \underline{H}_{RW} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T T_{RW}$$

The reaction wheel spin rate in (rad/sec) is

$$w_{RW} = \frac{H_{RW}(1)}{I_w}$$

The CMG rate of change of momentum is a function of the internal CMG torque ($\underline{T}_{CMG}$). The control torque experienced by the spacecraft in body axes due to gimbaling (δ) is ($-\dot{H}_{cmg}$), which is the rate of change in the CMG momentum.

$$\underline{\dot{H}}_{CMG} + \underline{\omega} \times \underline{H}_{CMG} = -\underline{T}_{CMG}$$

$$-\underline{\dot{H}}_{CMG} = \underline{T}_{con} = -\left[A(\delta)\right]\underline{\dot{\delta}}$$

The (2x3) matrix [A] consists of two column vectors ($\underline{a}_i$) which are functions of the gimbal angles (δ_i). They are also functions of the CMG quad and reference directions (q_i and r_i).

$$A(\delta) = \left(\underline{a}_1 \quad \underline{a}_2\right) \; where: \; \underline{a}_i = \left(\underline{q}_i \cos\delta_i - \underline{r}_i \sin\delta_i\right) h_{CMG(i)}$$

The two CMG gimbal directions are in pitch and yaw:

$$m = \begin{bmatrix}0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix}$$

The two CMG momentum reference directions (initially facing in opposite x directions at zero gimbal angles) are:

$$r = \begin{bmatrix} 1 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The quad directions from the cross-product ($\underline{m}_i$ x $\underline{r}_i$) are:

$$q = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}$$

The CMG gimbal rates are controlled by a servo system that generates gimbal torques. The servo torques are attempting to counteract the gyroscopic disturbance torques created by the spacecraft rate. $\dot{\theta}$ *and* $\dot{\phi}$ are the spacecraft rates resolved about the CMG reference and quad axes, as defined in equations (2.3). The CMG gimbal inertial acceleration is obtained by integrating the gimbal moment equation below, ignoring friction. T_{gi} is the motor torque applied at the gimbal. Even though the CMG moment of inertia about the gimbal J_g is relatively small, the gyroscopic moment caused by the CMG momentum h_{cmg} coupling with spacecraft rate is a big torque, requiring a powerful gimbal servo-motor in order to control the gimbal rate.

$$J_g \ddot{\delta}_i + h_{cmg(i)} \left(\dot{\theta} \sin\delta - \dot{\phi} \cos\delta \right) = T_{gi}$$

We also check the simulation by calculating the system momentum which is always constant. In this case it's zero because it is initialized at zero.

$$\underline{H}_{sys} = J_{sc}\,\underline{\omega} + \underline{H}_{CMG} + \underline{H}_{RW} = const.$$

This dynamic model is implemented in Matlab function "*SC_RW_2CMG_Dyn2.m*". There is also a simple dynamic model implemented in Matlab function "*SC_RW_2CMG-Dyn1.m*" that has simplified gimbal dynamics. The gimbal torques are not calculated in this model, but the gimbal rates are assumed to follow the commanded rates. The servo system is replaced by a 12 Hz second order CMG gimbal rate control model. These dynamic models are used in separate simulations. The attitude quaternion is updated using body rate with flexibility included (w_t= w + w_f). Flex rate at the gyro (w_f) is provided by the flex state-state model that gets excited by the MCS torque.

```
function xdot= SC_RW_2CMG_Dyn2(x,Tci,Tw,Td,wf)
global nc d2r r2d
global J Jinv Iw Jsi Jgi Joi hcmg
%-------------------------------------------------------------
% State Variables (x)
% x(1-3)   = Body rates   (w)   (rad/sec)
% x(4-7)   = Quaternion
% x(8:10)  = h (CMG momentum)
% x(11:13) = RW Momentum
% x(14:15) = deldot
% x(16:17) = delta
% Inputs:
% Tci(2)   = Gimbal Torques (ft-lb)
% Tw(1)    = Reaction Wheel Torque in spacrft x-axis, (ft-lb)
% Td(3)    = Disturbance Torque (ft-lb)
% wf(3)    = Flex rate only from FEM system
%-------------------------------------------------------------
xdot= zeros(30,1);
w=  x(1:3);                                          % S/C Body rates (rad/sec)
qt= x(4:7);                                          % S/C Quaternion Attitude
h = x(8:10);                                         % CMG Momentum
hrw= x(11:13);                                       % React Wheel Momentum (body)
deldot= x(14:15); delidot=deldot;                    % Gimbal Rates relatv to s/c
delta = x(16:17);                                    % Gimbal Angles (rad)

wt= w+wf;                                            % Total body rate + flex
wr = hrw(1)/Iw;                                      % React Wheel Rate (rad/sec)
Twi= [Tw, 0, 0]';                                    % Wheel Torque Vector (ft-lb)
hs = J*w + h + hrw;                                  % System Momentum (hs)
[Pj,thd,phd,ddd]= Transforms(delta*r2d,w);           % SC Rates al ref, quad, gmb

Tcmg=zeros(3,1);                                     % Calc CMG Torque in Body
for i=1:nc
    sd=sin(delta(i)); cd=cos(delta(i));
    Mj= [Tci(i); ...                                 % CMG Torque in CMG axis
    deldot(i)*((Jsi-Jgi)*(thd(i)*cd+phd(i)*sd) +hcmg(i)); ...
    deldot(i)* (Jgi-Joi)*(phd(i)*cd-thd(i)*sd)];
    Tcmg= Tcmg - Pj(:,:,i)*Mj;                       % Torque on Vehicle
    delidot(i)= delidot(i) + ddd(i);                 % Delta_Inertial_dot (rad)
end

xdot(1:3)= Jinv*(Tcmg +Twi - cross(w,J*w) );         % Vehicle acceleration
xdot(4:7)= 0.5*[0     wt(3)  -wt(2)  wt(1);          % Quaternion Update
             -wt(3)   0       wt(1)  wt(2);          % includes flex)
              wt(2)  -wt(1)   0      wt(3);
             -wt(1)  -wt(2)  -wt(3)  0]* qt;
xdot(8:10)= -Tcmg - cross(w,h);                      % Hcmg_dot
xdot(11:13)= -Twi - cross(w,hrw);                    % R-Wheel Momentum dot
for i=1:nc
    sd=sin(delta(i)); cd=cos(delta(i));
    xdot(13+i)= (Tci(i)-hcmg(i)* ...                 % Gimbal acceler delta-ddot
        (thd(i)*sd-phd(i)*cd))/Jgi;
    xdot(15+i)= x(13+i);                             % Gimbal rates  delta-dot
end

% Additional Outputs
xdot(18:20)= hs;                                     % System Momentum
xdot(21:23)= wt;                                     % Total Vehi rate
xdot(24:26)= h + hrw;                                % CMG + RW Momentum
xdot(27:29)= Tcmg + Twi;                             % CMG + RW Torques on s/c
xdot(30)   = wr;                                     % Wheel Rate (rad/sec)
```

Attitude Control System

The attitude control system is shown in Figure (3.6.2). The quaternion command (Q_{com}) is synthesized from the specified rotation angle and the eigenaxis command unit vector (com_dir). The quaternion command is compared with the spacecraft attitude quaternion feedback (Q_{fb}). The quaternion error (qe) is calculated using function "*qerror2.m*", and it is an input to the max energy attitude control function "*MaxEn_ACS.m*". The output is a body rate command (w_c), in roll, pitch, and yaw, which is an input to the steering logic. The ACS is a dual mode attitude control system. It uses a non-linear phase-plane logic when the attitude errors are large and it switches to PID control when the attitude error becomes small. Only the (PI) control gains are included in the ACS function. The rate feedback (D) part is implemented in the steering function. The switching criterion between phase-plane and PID control operations (qwm) is a combination of attitude magnitude plus rate magnitude. When it becomes sufficiently small it triggers the switch (kay) from zero to one which turns on the integrators that further improve command tracking performance. The control mode switching logic is implemented by means of an integration feedback loop. The ACS implementation is different in pitch and yaw than it is in roll using different bandwidths and PID gains. The phase-plane parameters are also different because the roll axis response is expected to be more sluggish than pitch and yaw which determine the Line-of-Sight (LOS) pointing performance.

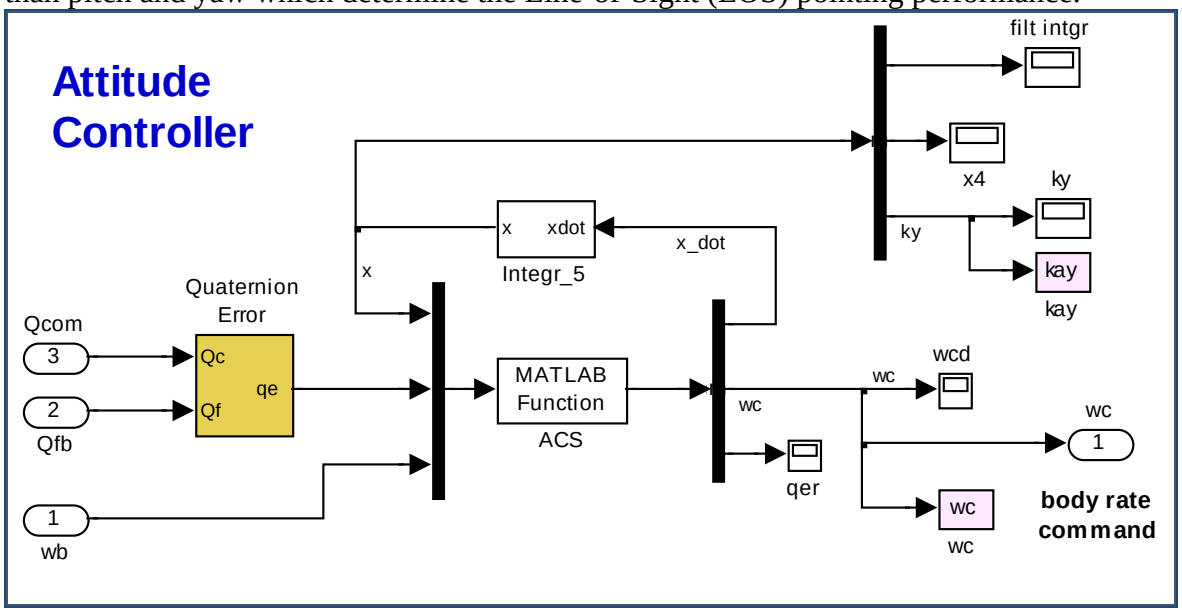


Figure 3.6.2 Max Energy Attitude Control System uses the function "*MaxEn_ACS.m*"

```
function dot= MaxEn_ACS(x,qe,wb)
global Ix Iy Iz d2r r2d
global Tmax maxvel maxvl1
% PID Attitude Controller with Energy and Velocity Limits
% Energy Manager and Maximum Velocity Limit for Large Errors
% Linear PID Controller for Small Errors
% ------------------------------------------------------
% State Variables (x)
% x(1-3)  = Position Integration
% x(4)    = Ky Integrator
% x(5)    = Ky Filter
% Inputs:
% qe(3)    = Attitude Error
% wb(3)    = Vehicle Body rates
```

```
% Outputs:
% dot(6:8) = wc (veloc command)
% dot(9:11)= [ky,angle,qwm]
% ------------------------------------------------------
dot= zeros(11,1);                                   % 11 outputs
Jd= [Ix, Iy, Iz];                                   % Ixx Iyy Izz (slug-ft^2)
tlim=[10.0, Tmax,Tmax];                             % Max Torques x,y,z
mx=tlim./Jd;                                        % Max accelerations
ki=0.08; kp=3.0; ki2=0.012; kp2=0.3;                % PI Gains
a=asin(sqrt(qe'*qe)); angle=a*2*r2d;                % Maneuver error angle
wbm=sqrt(wb'*wb)*r2d;                               % Rate Error magnitude
qwm= 0.3*(angle + wbm*3);                           % attit + rate error magn

q1=qe(1); qq=qe(2:3);                               % Split [ptch,yaw]&[roll]
ky=(1.0-x(5));                                      % x(5)= (from 1 to 0)
dot(1) = ki2*q1 - (ky*ki2/kp2)*(x(1) +kp2*q1);      % Roll Posit Integrat.
dot(2:3)= ki*qq - (ky*ki/kp)*(x(2:3) +kp*qq);       % Ptch/Yaw Posit Integrat
wccc(1)  = kp2*q1 + x(1);                           % PI controller (x)
wccc(2:3)= kp*qq + x(2:3);                          % PI controller (y,z)
wcc = zeros(3,1); wc=wcc;                           % Initialize

% Energy Limit ----------------------------------------
if (qq'*qq)==0; alim=0; wclim=[0;0];
else;   alim=1/sqrt((qe(2)/mx(2))^2+(qe(3)/mx(3))^2);
        wclim= sqrt(2*qq.*qq*alim); end

if q1==0; alim1=0; wclm1=0;
else;     alim1= abs(mx(1)/q1); wclm1= sqrt(2*q1*q1*alim1); end

for i=1:2                                                % The 2 SG-CMGs
    if abs(wccc(i+1))>=wclim(i) & sqrt(qq'*qq)>0.00001 & abs(wccc(i+1))>0
           wcc(i+1)= wccc(i+1)*wclim(i)/abs(wccc(i+1));
    else;  wcc(i+1)= wccc(i+1); end
end

if abs(wccc(1))>=wclm1 & abs(q1)>0.00001 & abs(wccc(1))>0  % The RW
          wcc(1)= wccc(1)*wclm1/abs(wccc(1));
else;     wcc(1)= wccc(1); end

% Velocity Limit --------------------------------------
if abs(wcc(1))>maxvl1; wc(1)=maxvl1*q1/abs(q1); else; wc(1)= wcc(1); end
if sqrt(wcc(2:3)'*wcc(2:3))>maxvel; wc([2:3],1)=maxvel*qq/sqrt(qq'*qq);
else; wc([2:3],1)=wcc([2:3],1); end
dot(6:8)= wc;

% Compute ky ------------------------------------------
del=0; if qwm<1; del=1; end
dot(4) = del*2;
if x(4)>1; dot(4)= dot(4) + (1-x(4))*4; end
if x(4)<0; dot(4)= dot(4) - x(4)*4; end
kyin = max(0,min(1.0,x(4)));
dot(5)= (kyin-x(5))*4;
dot(9:11)= [ky,angle,qwm]';
```

CMG and RW Steering

The Momentum Control System (MCS) steering law is the inner rate loop in the attitude control system that controls the spacecraft rate by issuing gimbal rate commands to the CMGs and a torque command to the reaction wheel. Figure (3.6.3) shows the Simulink diagram of the steering subsystem which is implemented in function "*Steering_RW-CMG.m*".

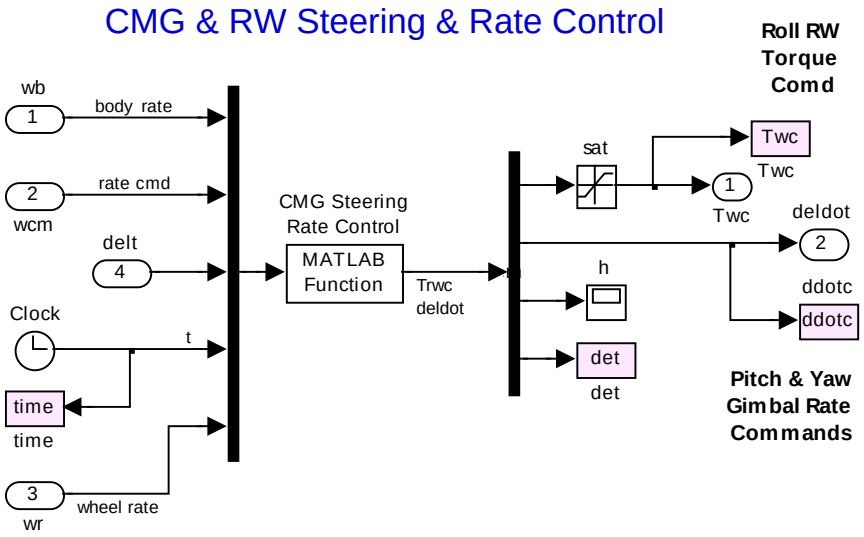


Figure 3.6.3 Steering and Singularity Avoidance System using the function "Steering_RW_CMG.m"

The input is spacecraft rate error (w_e) which is rate command minus body rate. The rate errors are converted to acceleration commands ($\dot{w}_{com}$) by multiplying them with rate gains, which are different. The roll gain is lower than pitch and yaw because of reduced bandwidth and control authority in roll. The acceleration commands ($\dot{w}_{com}$) are limited by software limits that prevent excessive torque demands. The CMG gimbal angles are also inputs to the steering control law. They are used for calculating the gimbal rate commands and an estimate of the CMG momentum from the following equation.

$$\underline{H}_{CMG} = \sum_{i=1}^{2} \left(\underline{r}_i \cos\delta_i + \underline{q}_i \sin\delta_i \right) h_{CMG_{(i)}}$$

The input torque ($\underline{T}_{CMG}$) is the internal control torque provided by the CMGs and it consists of the commanded control torque (T_{con}) plus an estimate of the gyroscopic torque $\omega \times H_{cmg}$. The control torque is the CMG steering logic $-[A(\delta)]\dot{\underline{\delta}}$ which is a function of the gimbal rates.

$$\underline{T}_{CMG} = \underline{T}_{con} - \left(\underline{\omega} \times \underline{H}_{CMG} \right)_{estim}$$

$$\underline{T}_{CMG} = -[A(\delta)]\dot{\underline{\delta}} - \left(\underline{\omega} \times \underline{H}_{CMG} \right)_{estim}$$

The CMG gimbal rate commands are calculated from the following equation where the second term uses an estimate of the system momentum to counteract the gyroscopic effects due to accumulated system momentum.

$$\underline{\dot{\delta}}_{com} = -\left\{\left[A^T A + \lambda E\right]^{-1} A^T\right\}\left[J_{sc}\dot{w}_{com} + \left(\omega \times H_{sys}\right)_{estim}\right]$$

Where $\underline{\dot{w}}_{com} = \begin{pmatrix} 0 & \dot{w}_{com(2)} & \dot{w}_{com(3)} \end{pmatrix}^T$ consists of only the pitch and yaw elements of the body rate commands.

After substituting $\dot{\delta} = \dot{\delta}_{com}$ in the spacecraft dynamics equations we obtain the following relationships, meaning, that the rate of change of CMG momentum is equal to the control torque, and that the spacecraft rate is equal to the commanded rate.

$$\dot{\underline{\omega}} = \underline{\dot{w}}_{com}$$

$$\underline{\dot{H}}_{CMG} = \left[A(\delta)\right]\underline{\dot{\delta}} = -\underline{T}_{con} = -\left[J_{sc}\underline{\dot{w}}_{com} + \left(\underline{\omega} \times \underline{H}_{sys}\right)_{estim}\right]$$

The (2x3) matrix [A] consists of two column vectors ($\underline{a}_i$) which are functions of the gimbal angles (δ_i). They are also functions of the CMG quad and reference directions as shown below.

$$A(\delta) = \begin{pmatrix} \underline{a}_1 & \underline{a}_2 \end{pmatrix} where: \underline{a}_i = \left(\underline{q}_i \cos\delta_i - \underline{r}_i \sin\delta_i\right) h_{CMG(i)}$$

There is also a gimbal rate command limit (wc_{lim}). The term (λE) is used for avoiding singularities (gimbal locks) in the pseudo-inversion of matrix A(δ).

$$E = \begin{bmatrix} 1 & 0.01\sin(0.2t) \\ 0.01\sin(0.2t) & 1 \end{bmatrix}; \; and \; \lambda = 10^6 \Big/ \det(A^T A)$$

The estimated system momentum consists of spacecraft plus CMG plus reaction wheel estimated momentum.

$$\underline{H}_{sys} = J_{sc}\underline{\omega} + \underline{H}_{CMG} + \underline{H}_{RW}$$

The reaction wheel torque command (T_{RW}), which controls roll, is calculated from the following equation. The second term is counteracting the gyroscopic torque generated in roll due to spacecraft rate coupling with system momentum. The torque command does not exceed ±10 (ft-lb).

$$T_{RW} = J_{sX}\,\dot{w}_{com(1)} - \left[\underline{\omega} \times \underline{H}_{RW}\right] \bullet \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$$

```
function out= Steering_RW_CMG(wb,wbc,delta,t,wr)
global m ref quad wcmg d2r r2d
global J Jinv Ix Iy Iz Iw hcmg
global Alim ddmax

% Inputs:
% wb(3)   = Body rates
% wbc(3)  = Body rate commands
% delta(2)= Gimbal angles
% t       = time
% wr(1)   = R-wheel rate (z)

krc= 5.2; krw=2.5;                                  % CMG, RW rate gains
out= zeros(7,1); h= [0 0 0]';                       % 7 outputs
we = wbc-wb;                                        % Rate error
wdot1= krw*we(1);   wdmg1=abs(wdot1);               % RW  Accelerat Command
wdot2= krc*we(2:3); wdotmag=sqrt(wdot2'*wdot2);     % CMG Accelerat Command

% Acceleration Limit
if wdmg1>0.008; wdot1= 0.008*wdot1/wdmg1; end       % Limit x accelerat
if wdotmag > Alim; wdot2= Alim*wdot2/wdotmag; end   % Limit y & z accelerat

for i=1:2
    A(:,i)=(-sin(delta(i))*ref(:,i) + cos(delta(i))*quad(:,i))*hcmg(i);
    h = h + (cos(delta(i))*ref(:,i) + sin(delta(i))*quad(:,i))*hcmg(i);
end

% Singularity Avoidance Logic
lamb= 1.0e6/det(A'*A);
E= [1               0.01*sin(0.2*t);
    0.01*sin(0.2*t)           1];
pinverse= inv(A'*A + lamb*E)*A';
hrw= [1 0 0]'*Iw*wr;                                % RW momentum
Hsys= J*wb + h + hrw;                               % System Momentum
Hmcs= h+hrw;                                        % Comb MCS Momentum

hdot = J*[0;wdot2];                                 % Desired torque in y,z
deldot= -pinverse*(hdot-cross(wb,Hsys));            % Delta dot command
ddmag=sqrt(deldot'*deldot);                         % Gimbal rate magnitude
if ddmag>ddmax                                      % Limit Gimbal rate
   deldot=ddmax*deldot/ddmag;
end

gyro= cross(wb,hrw);                                % Gyro torque resolved in the x axis
out(1)= Ix*wdot1 - gyro(1);                         % R-Wheel Torque command
out(2:3)= deldot;
out(4:6)= h;
out(7)= 1/lamb;
```

Analysis Models and Files

The files for this analysis are located in folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(d) Flex SC with 2SGCMG+RW ACS*". This folder contains the spacecraft with RW and CMG dynamics in file "*SC_RW_2CMG-Dyn2.m*", the simulation model in file "*MaxEn_RW_2SGCMG_Flex.mdl*", and the initialization file "*start.m*" that initializes the simulation parameters, such as, mass properties, CMG and reaction wheel parameters and axes, initial spacecraft attitude, rate, wheel rate, and CMG gimbal angles, acceleration and velocity limits in the attitude control system, torque limits, and the direction of the commanded maneuver which is a unit vector. There is also a simple simulation model that does not include detailed gimbal dynamics and structural flexibility in Simulink file "*MaxEn_RW_2SGCMG.mdl*" which uses the function "*SC_RW_2CMG_Dyn1.m*". This model is also initialized by "*start.m*". The CMG and RW steering law is in function "*Steering_RW_CMG.m*", and the attitude control law is in function "*MaxEn_ACS.m*". There are linearized versions of these functions in files "*Steering_Lin.m*" and "*ACS_Lin.m*" which are used for frequency response stability analysis by running file "*freq.m*". The structural flexibility state-space model "*flex_only_fem_s.m*" is combined with the rigid-body non-linear dynamics function "*SC_RW_2CMG_Dyn2.m*". It was calculated in section (1) and used in previous CMG simulations. It consists of only flex dynamics (no rigid-body modes). Although the flex model has many inputs and outputs we are only using the transfer function between the MCS torques at node #6 and the rate gyro sensors at node #2. There are also two plotting files "*pl1.m*" and "*pl2.m*" that plot the simulation results from the simple and detailed simulations respectively.

Simple Simulation Model

The simple Simulink model "*MaxEn_RW_2SGCMG.mdl*" is shown in Figure (3.6.4). The ACS is common to both models and was described earlier. The steering law is shown in Figure (3.6.5). It calculates the RW torque command and the CMG gimbal rate commands. The RW torque is limited to ±10 (ft-lb).

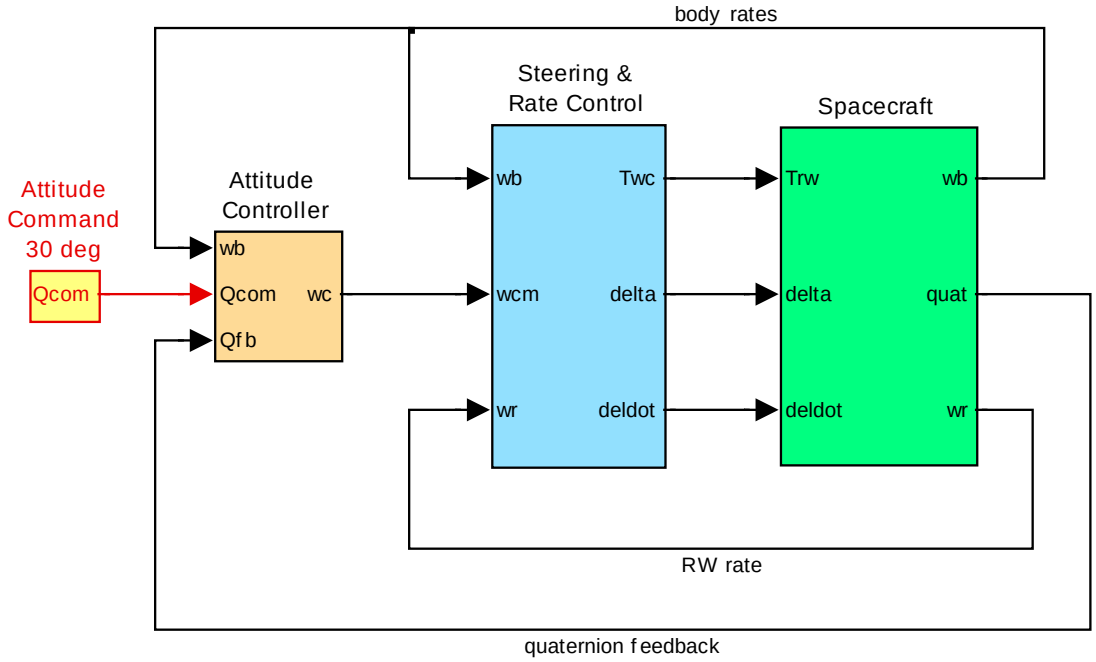


Figure 3.6.4 Simple Spacecraft model using Max Energy Control with one RW and two SG-CMG

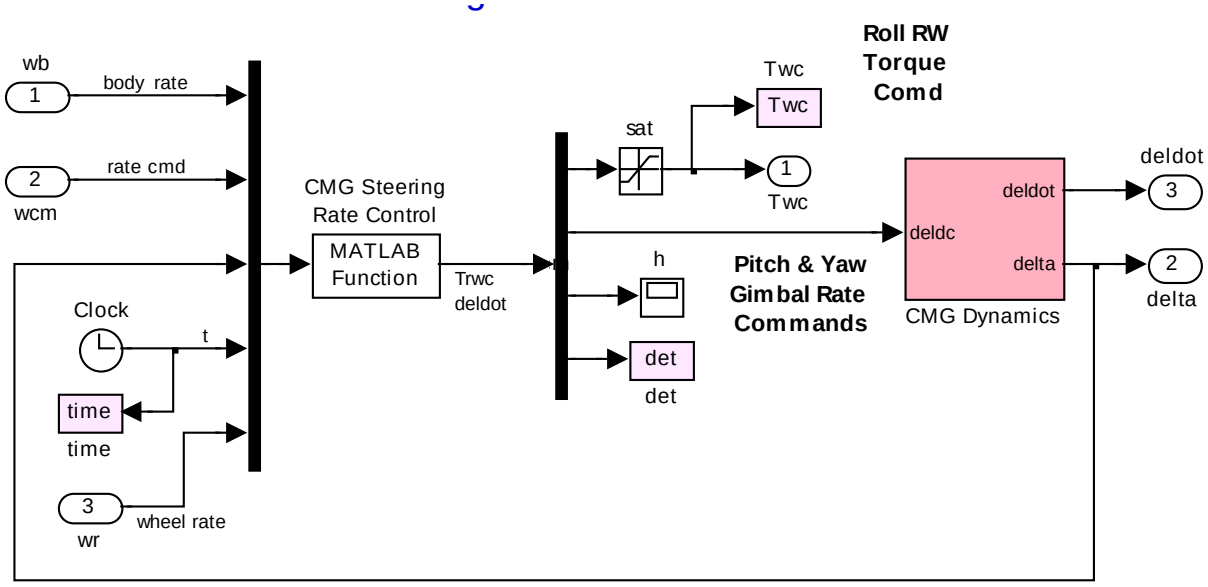


Figure 3.6.5 CMG and RW Steering

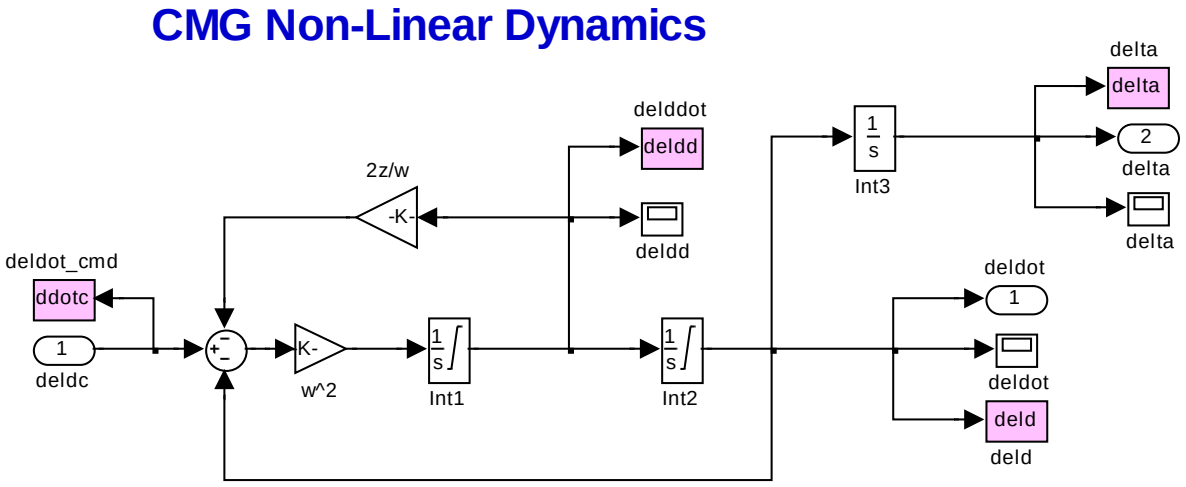


Figure 3.6.6 Second order CMG dynamics

Figure (3.6.6) shows the simplified 2nd order gimbal dynamics. There are no gimbal torques in this model. The inputs are pitch and yaw gimbal rate commands for the two CMGs, and the outputs are gimbal angles, rates, and gimbal accelerations. The integrators have limits (W_{clim} and A_{clim}) to bound the gimbal rates and accelerations respectively. The gains are a function of the CMG servo system bandwidth and damping coefficient (w_{cmg} and zeta) defined in the initialization file "start.m". The spacecraft dynamic model is shown in Figure (3.6.7). It is implemented by Matlab function "*SC_RW_2CMG-Dyn1.m*". The inputs are RW torque, gimbal rates and angles. The outputs are body rates, attitude quaternion, wheel rate, CMG and system momentum, and the combined MCS torque. The file "pl1.m" plots the results when the simulation is complete.

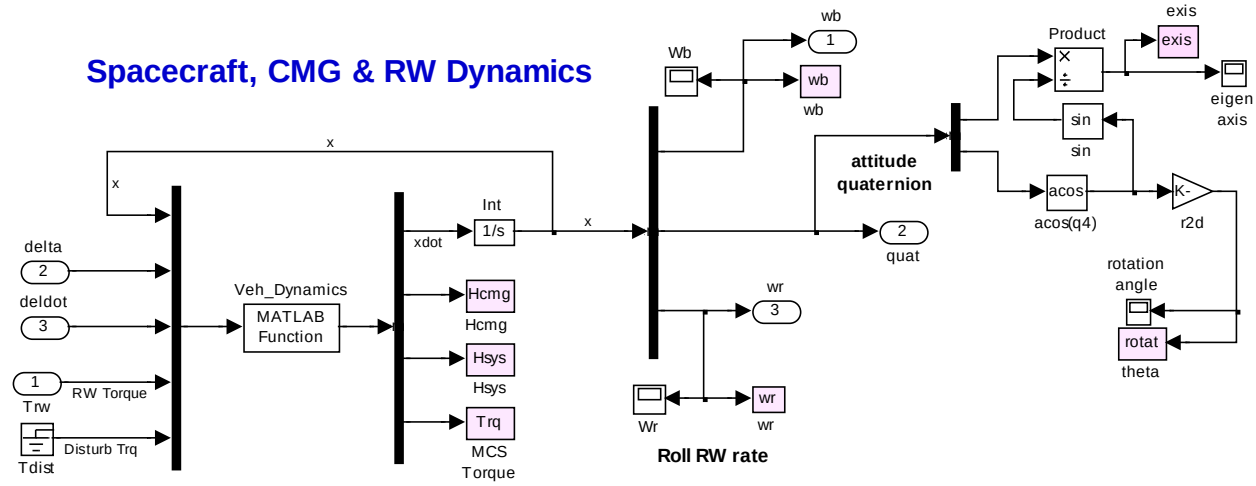


Figure 3.6.7 Spacecraft dynamic model uses the function "SC_RW_2CMG_Dyn1.m"

Simulation Model with Structural Flexibility and Gimbal Controls

The detailed simulation model is in file "*MaxEn_RW_2SGCMG_Flex.mdl*" and is shown in Figure (3.6.8). The ACS and Steering laws are the same as in the simple model and they were described earlier. The CMG dynamics, however, is not represented by a second order transfer function with rate and acceleration limits, but in this case we are modeling the gimbal rate control servo system which provides actual torques at the CMG gimbals. This model captures the gyroscopic torques at the gimbals due to the $\omega \times h$ effects. The RW torque mechanization is the same as before.

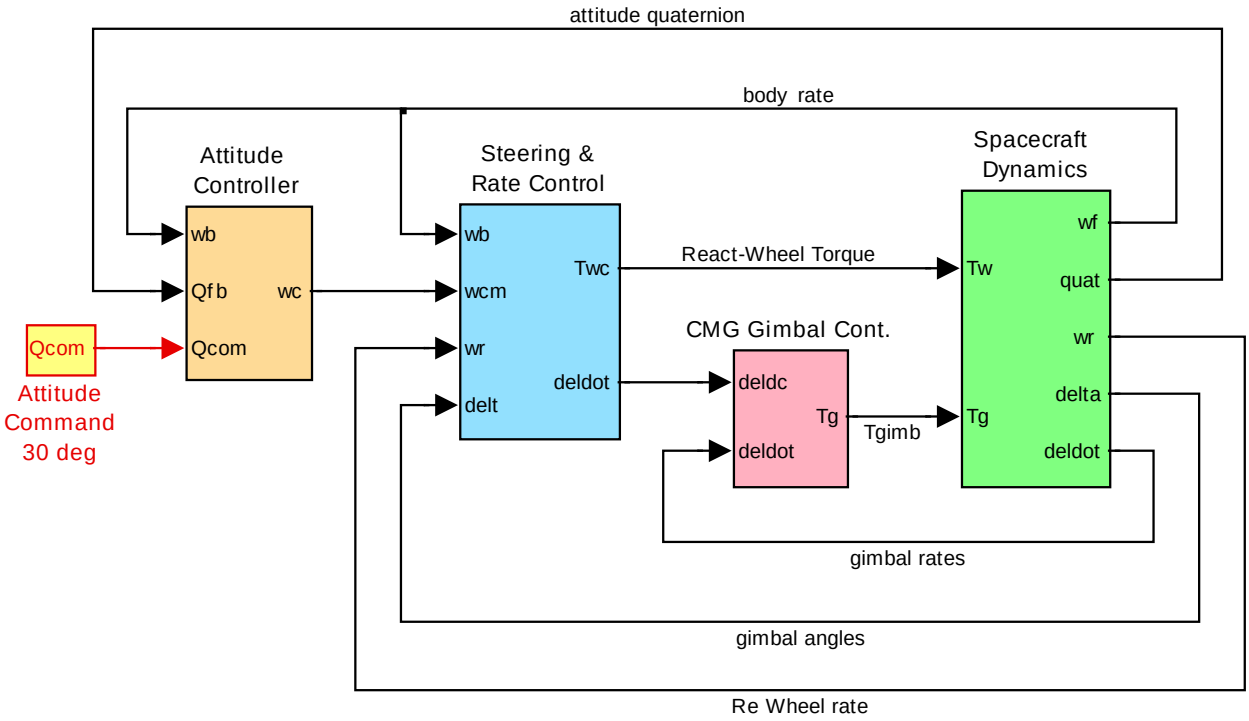


Figure 3.6.8 Spacecraft simulation model with structural flexibility and detailed gimbal dynamics included

Figure (3.6.9) shows the gimbal servo system that controls the gimbal rates. The inputs are gimbal rate commands and gimbal rate measurements from the CMG resolver. The gimbal rotational dynamics are included in the spacecraft dynamics block and will be discussed later. The outputs are pitch and yaw gimbal torques (T_{gi}) for the two CMGs. It is basically a (PI) controller where its proportional and integral gains are determined from the control system bandwidth. The bandwidth is set high (35 Hz) because the servo has to fight the gyroscopic torques which are not small. There is a torque limit that limits the gimbal torque because the servo motor has limited torque capability.

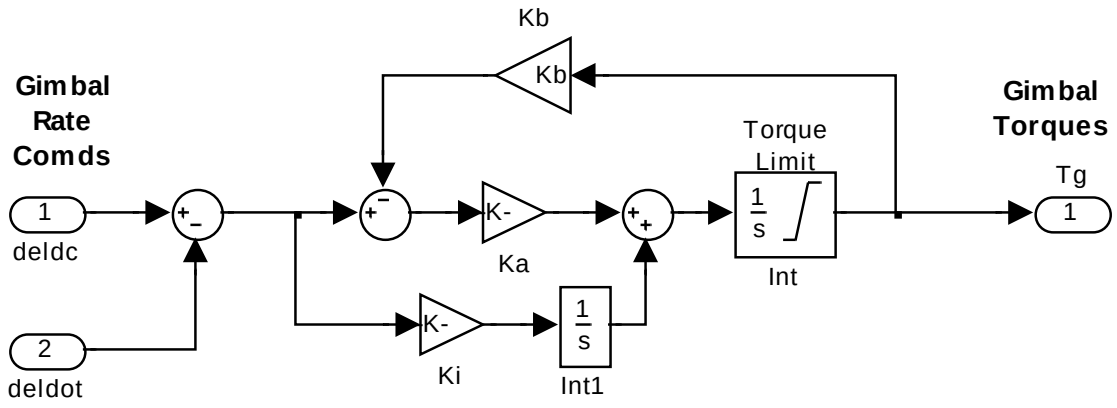


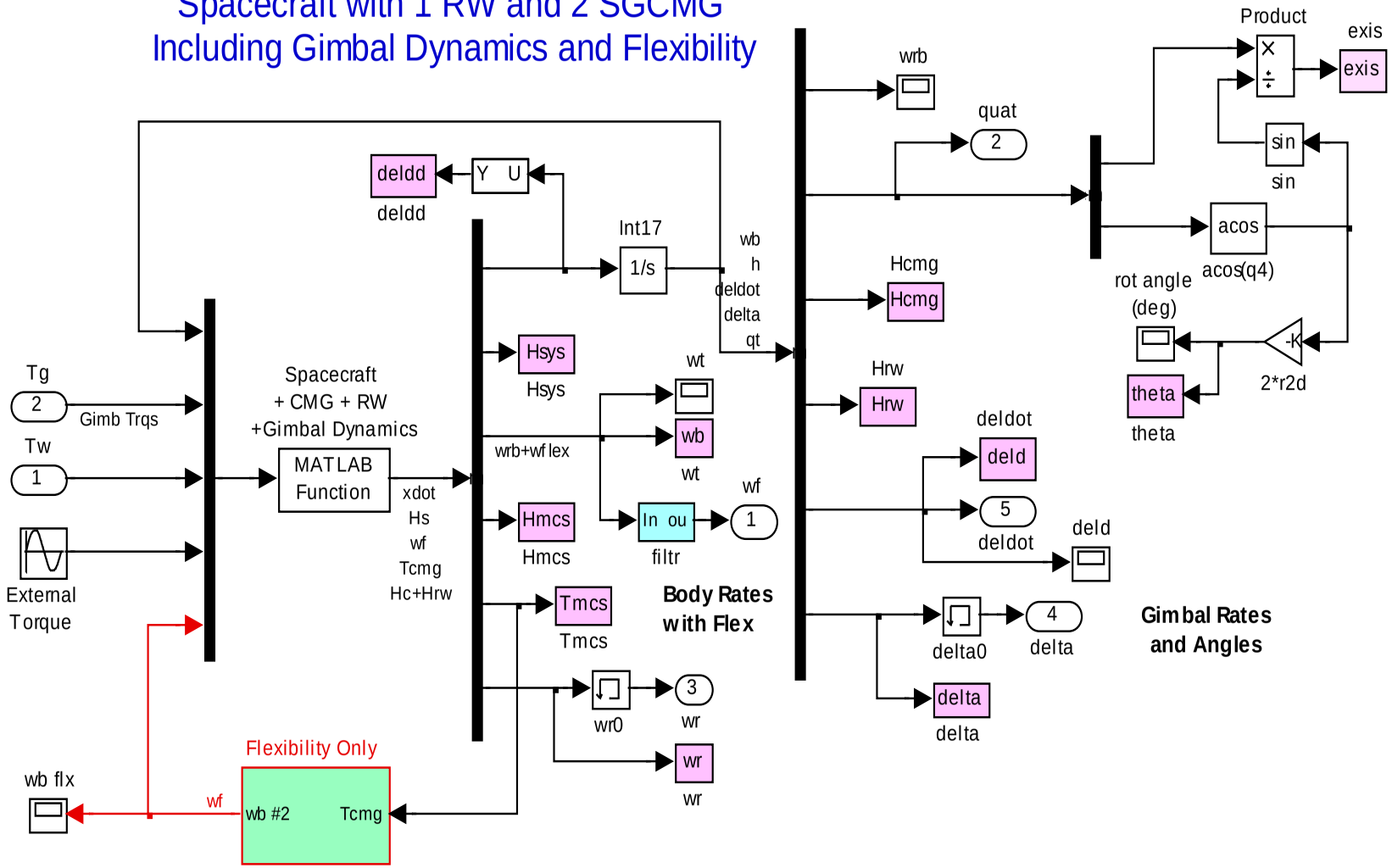
Figure 3.6.9 CMG gimbal rate control system

The spacecraft dynamics block is shown in detail in Figure (3.6.10). The spacecraft, CMG and RW dynamics is implemented in Matlab function "*SC_RW_2CMG_Dyn2.m* ", shown below. The state-vector consists of 17 states (body rates, quaternion, CMG momentum, RW momentum, gimbal rates, and gimbal angles). The state derivatives ($\underline{x}_{dot}$) are generated by the spacecraft function, they integrated externally, and they are fed back as input states ($\underline{x}$). The remaining inputs are: CMG gimbal torques (T_{gi}), RW torque (T_w), disturbance torque (T_d), and flex rate (w_f) from the structural flexibility model. A combined rigid-body plus flex rate (w_t) is generated by adding the rigid-body (w) and the flex rates (w_f) together. The attitude quaternion is propagated using the combined rate (w_t). The RW angular rate (h_{rw}) is computed from the RW momentum. The system momentum (hs) consists of spacecraft plus CMG plus RW momentum. It is calculated to make sure it remains constant when the external disturbances are zero.

The function "*Transforms*" resolves the spacecraft rates along the CMG (r_i, q_i, and m_i) directions based on the gimbal angles (δ_i), see equation (2.3) in Section 2 (Spacecraft with Single Gimbal CMG). It computes also the projection matrix [Pj], equation (2.5) which is used to calculate the CMG torques in the spacecraft axes. The dynamics program continues to calculate the torque applied on each CMG along the three reference axes, that is, gimbal, spin, and output axes. The torques are converted using matrix Pj, to total CMG torque in spacecraft axes (T_{cmg}). The gimbal accelerations $\underline{x}_{dot(14:15)}$ are computed as a function of the applied torques. Each gimbal experiences two torques, the torque applied by the servo motor T_{gi}, and a gyroscopic torque as described in equation (2.16). The motor torques are constantly trying to fight the gyroscopic torques in order to control the gimbal rates as dictated by the steering law. The dynamics function also calculates the combined CMG plus RW torque in body axes ($\underline{T}_{mcs}$), and the combined CMG plus RW momentum ($\underline{H}_{mcs}$) in body axes. $\underline{T}_{mcs}$ is used as an input to the flex model to excite the structural dynamics. The flex dynamics is a state-space model "*Flex Only Spacecraft with RCS and CMG*" which is read from function "*flex_only_fem_s.m*" and it was used in earlier examples using 4 CMGs. It contains only flex modes (no rigid modes). Its output (w_f) loops back and becomes an input to the spacecraft model, as shown in Figure (3.6.10). A filter is used in the out-of-plane combined rate (w_t) to attenuate a flex mode at 14 Hz.

```
function xdot= SC_RW_2CMG_Dyn2(x,Tci,Tw,Td,wf)
global nc d2r r2d
global J Jinv Iw Jsi Jgi Joi hcmg
%-------------------------------------------------------------
% State Variables (x)
% x(1-3)   = Body rates   (w)   (rad/sec)
% x(4-7)   = Quaternion
% x(8:10)  = h (CMG momentum)
% x(11:13) = RW Momentum
% x(14:15) = deldot
% x(16:17) = delta
% Inputs:
% Tci(2)   = Gimbal Torques (ft-lb)
% Tw(1)    = Reaction Wheel Torque in spacrft x-axis, (ft-lb)
% Td(3)    = Disturbance Torque (ft-lb)
% wf(3)    = Flex rate only from FEM system
%-------------------------------------------------------------
xdot= zeros(30,1);
w=  x(1:3);                                              % S/C Body rates (rad/sec)
qt= x(4:7);                                              % S/C Quaternion Attitude
h = x(8:10);                                             % CMG Momentum
hrw= x(11:13);                                           % React Wheel Momentum (body)
deldot= x(14:15); delidot=deldot;                        % Gimbal Rates relatv to s/c
delta = x(16:17);                                        % Gimbal Angles (rad)

wt= w+wf;                                                % Total body rate + flex
wr = hrw(1)/Iw;                                          % Reaction Wheel Rate (rad/sec)
Twi= [Tw, 0, 0]';                                        % Wheel Torque Vector (ft-lb)
hs = J*w + h + hrw;                                      % System Momentum (hs)
[Pj,thd,phd,ddd]= Transforms(delta*r2d,w);               % SC rates al (ref,quad,gimb)

Tcmg=zeros(3,1);                                         % Calcul CMG Torque in Body
for i=1:nc
    sd=sin(delta(i)); cd=cos(delta(i));
    Mj= [Tci(i); ...                                     % CMG Torque in CMG axis
    deldot(i)*((Jsi-Jgi)*(thd(i)*cd+phd(i)*sd) +hcmg(i)); ...
    deldot(i)* (Jgi-Joi)*(phd(i)*cd-thd(i)*sd)];
    Tcmg= Tcmg - Pj(:,:,i)*Mj;                           % Torque on Vehicle
    delidot(i)= delidot(i) + ddd(i);                     % Delta_Inertial_dot (rad)
end

xdot(1:3)= Jinv*(Tcmg +Twi - cross(w,J*w) );             % Vehicle acceleration
xdot(4:7)= 0.5*[0     wt(3)  -wt(2)  wt(1);              % Quaternion Update
             -wt(3)   0       wt(1)  wt(2);              % includes flex)
              wt(2)  -wt(1)   0      wt(3);
             -wt(1)  -wt(2)  -wt(3)  0]* qt;
xdot(8:10)= -Tcmg - cross(w,h);                          % Hcmg_dot
xdot(11:13)= -Twi - cross(w,hrw);                        % R-Wheel Momentum dot
for i=1:nc
    sd=sin(delta(i)); cd=cos(delta(i));
    xdot(13+i)= (Tci(i)-hcmg(i)* ...                     % Gimbal acceler delta-ddot
        (thd(i)*sd-phd(i)*cd))/Jgi;
    xdot(15+i)= x(13+i);                                 % Gimbal rates  delta-dot
end

% Additional Outputs
xdot(18:20)= hs;                                         % System Momentum
xdot(21:23)= wt;                                         % Total Vehi rate
xdot(24:26)= h + hrw;                                    % CMG + RW Momentum
xdot(27:29)= Tcmg + Twi;                                 % CMG + RW Torques on s/c
xdot(30)   = wr;                                         % Wheel Rate (rad/sec)
```

Spacecraft with 1 RW and 2 SGCMG Including Gimbal Dynamics and Flexibility



The following file "*start.m*" initializes both simulation models.

```
global m ref quad wcmg zeta bet gam nc d2r r2d
global J Jinv Ix Iy Iz Iw Jsi Jgi Joi hcmg
global Hmax Tmax Alim Jlim maxvel maxvl1 Aclim Wclim
d2r= pi/180; r2d= 180/pi;
[Af, Bf, Cf, Df] = flex_only_fem_s;               % Load Flex Only Spacecr Dynamics
Ctr= [0 1 0; 1 0 0; 0 0 -1];                      % Appendage Transform Matric

J= [0.17E+94, -0.16e+93,  0.11E+92;               % Vehicle MOI matrix in (slug-ft^2).
   -0.16e+93,  1.30E+94,  0.31E+92;
    0.11E+92,  0.31E+92,  1.41E+94];
Jinv=inv(J); Ix=J(1,1); Iy=J(2,2); Iz=J(3,3);

% CMG data
m =  [0   0;                                      % CMG Gimbal Directions
      1   0;                                      % Pitch Gimbal
      0   1];                                     % Yaw Gimbal

ref= [1  -1;                                      % CMG Init Momentum Directions
      0   0;                                      % Initially spin vectors are ...
      0   0];                                     % in roll direction back to back

for i=1:2;
    quad(:,i)=cross(m(:,i),ref(:,i));             % 3rd Orthogonal axis
end

bet= [-90,0]'; gam=[0,-180]; nc=2;                % CMG mounting angles
wc=250; z= 0.98; lamb=0.4;                        % CMG servo bandw (rad/s), damp coeff
Jsi= 1.2; Jgi= 0.6; Joi= 0.8;                     % CMG Inertia Spin, Gimb, Outp axes
Kii=lamb*Jgi*wc^3;                                % CMG Servo Gains
Ka=(1+2*z*lamb)*Jgi*wc^2;
Kb=(2*z+lamb)*wc/Ka;

% RW data
Hm= 350;                                          % RW Momentum 150 (ft-lb-sec)
Wm= 3500*2*pi/60;                                 % Max RW Speed 2500*2*pi/60 (rad/sec)
Iw= Hm/Wm;                                        % Wheel rotor inertia from 4=Iw*Wmax
Twmax=10;                                         % Max R-Wheel Torque

% Initialization data
wb0= [0  0  0]*d2r;                               % Initial body rates
Qt0= [0  0  0  1];                                % Initial Quaternion
h0 = [0  0  0];                                   % Initial CMG Momentum (body)
hrw0=[0  0  0];                                   % Initial React-Wheel Momentum
deldot0= [0 0]*d2r;                               % Initial Gimb Rates
delta0 = [0 0]*d2r;                               % Initial Gimb Angles
wr0=0;                                            % Initial RW rate
ini = [wb0,Qt0,h0,hrw0,deldot0,delta0];           % State Integrator Initialization
ini2= [wb0,Qt0,wr0];                              % Old Sim Initialization

Tglim=40;
Aclim=200*d2r; Wclim=60*d2r;                      % CMG Model Gimbal Acceler/Rate Limits
Alim= 0.022; Jlim=120*d2r;                        % Max Steering Accel 0.024 (rad/s^2)
maxvel=0.04; maxvl1=0.025;                        % ACS Veloc limits(rad/sec)
Tmax=250;                                         % Max Torque 300
Hmax=1200; hcmg=[Hmax, Hmax];                     % Max CMG Momentum, Momentum Array
wcmg= 80; zeta= 0.9;                              % CMG bandwidth (rad/s), damping coeff

com_dir=[0; 1; 1];                                % Command Direction Unit Vector
com_dir=com_dir/sqrt(com_dir'*com_dir);
```

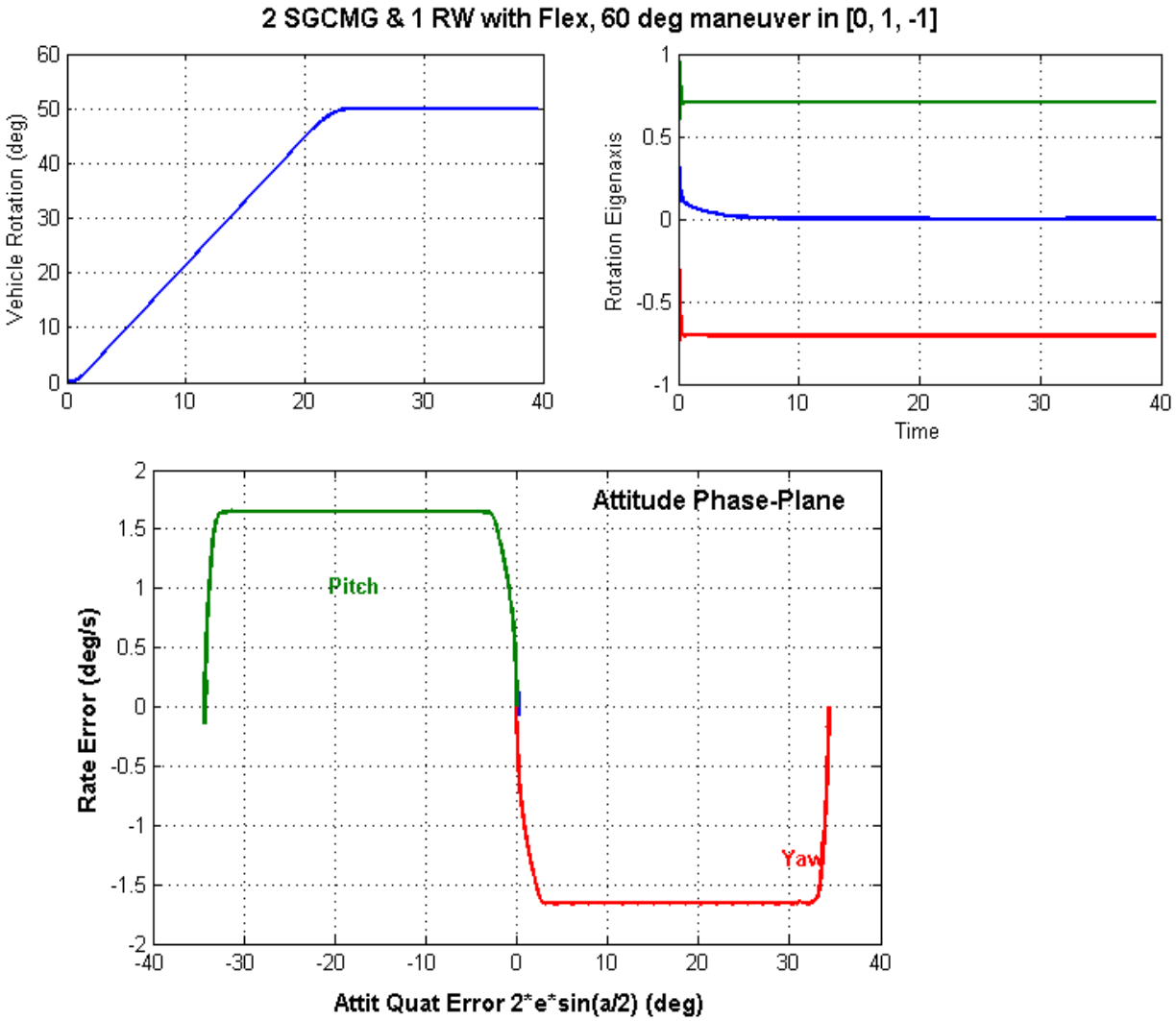
Simulation Results

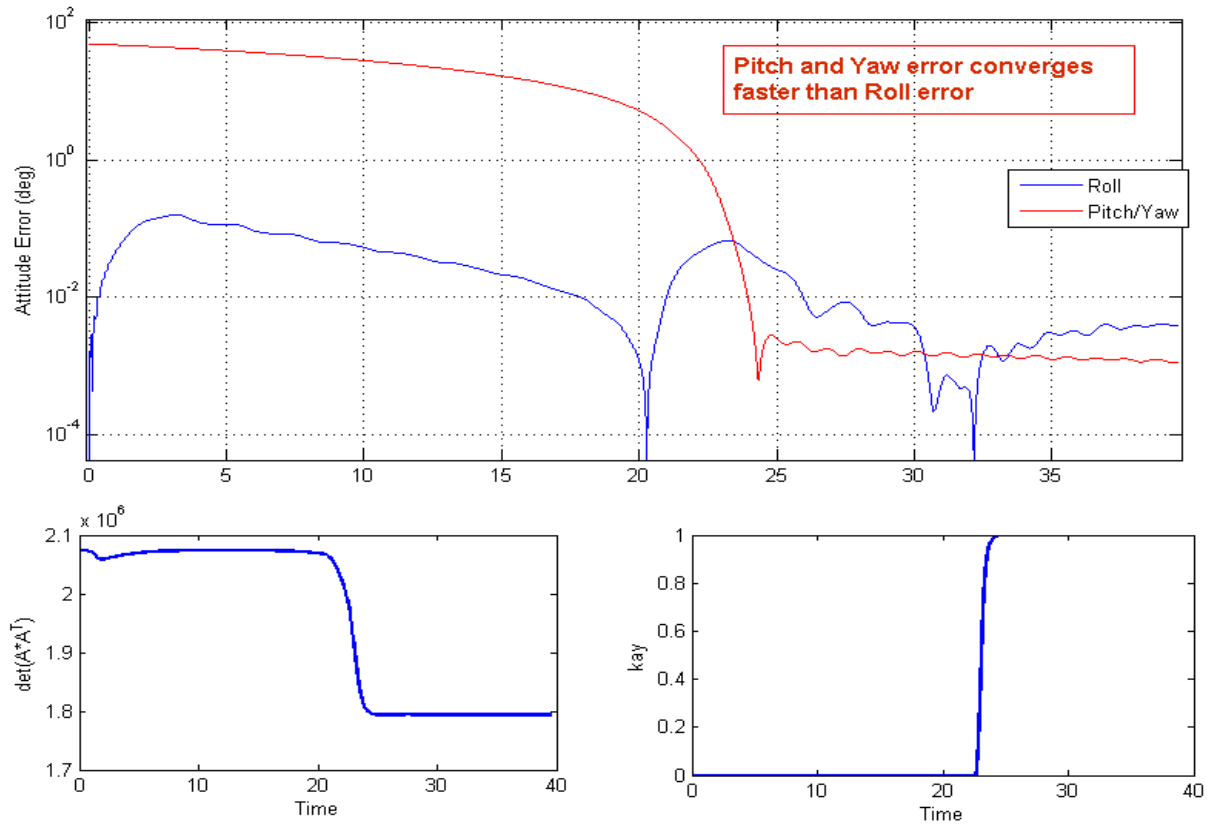
When you combine CMGs and RW momentum exchange devices you have to take into consideration that the system is not “equally yoked” by design. In this spacecraft the CMGs are dedicated to control pitch and yaw (which are more important), and the reaction wheel is dedicated to control roll (where longer conversion times are acceptable). In general the maneuvers are in pitch and yaw. We don’t command roll. The CMGs, however, do not only produce pitch and yaw torques but they also produce a roll torque which is a disturbance to the reaction wheel roll control system. The wheel has to accelerate in order to take the disturbance out, but the wheel has very limited torque and momentum capability.

The CMG generated roll disturbances are small when the gimbal angles are small. When the spacecraft performs small angle maneuvers around Nadir the CMG gimbal angles are small and the induced roll torque is also small, so the RW control system easily compensates against the roll CMG torques. But when the maneuvers become larger than 40º the RW control system sometimes has difficulty overcoming the CMG roll torque and the maneuver may fail. Let us remind you that this is a low budget design, in comparison to the previous 4 SGCMG design, because the spacecraft pointing requirement is not greater than 40º from Nadir (earth center) direction. The CMG/RW momentum control system combination will perform better in some maneuvers than others, depending on how big is the induced roll disturbance by the CMGs. We will present and discuss simulation results obtained from five separate maneuvers. The roll responses may be a little sloppy most of the time but overall the pitch and yaw responses are great. The results obtained include vehicle flexibility as described in the simulation models.

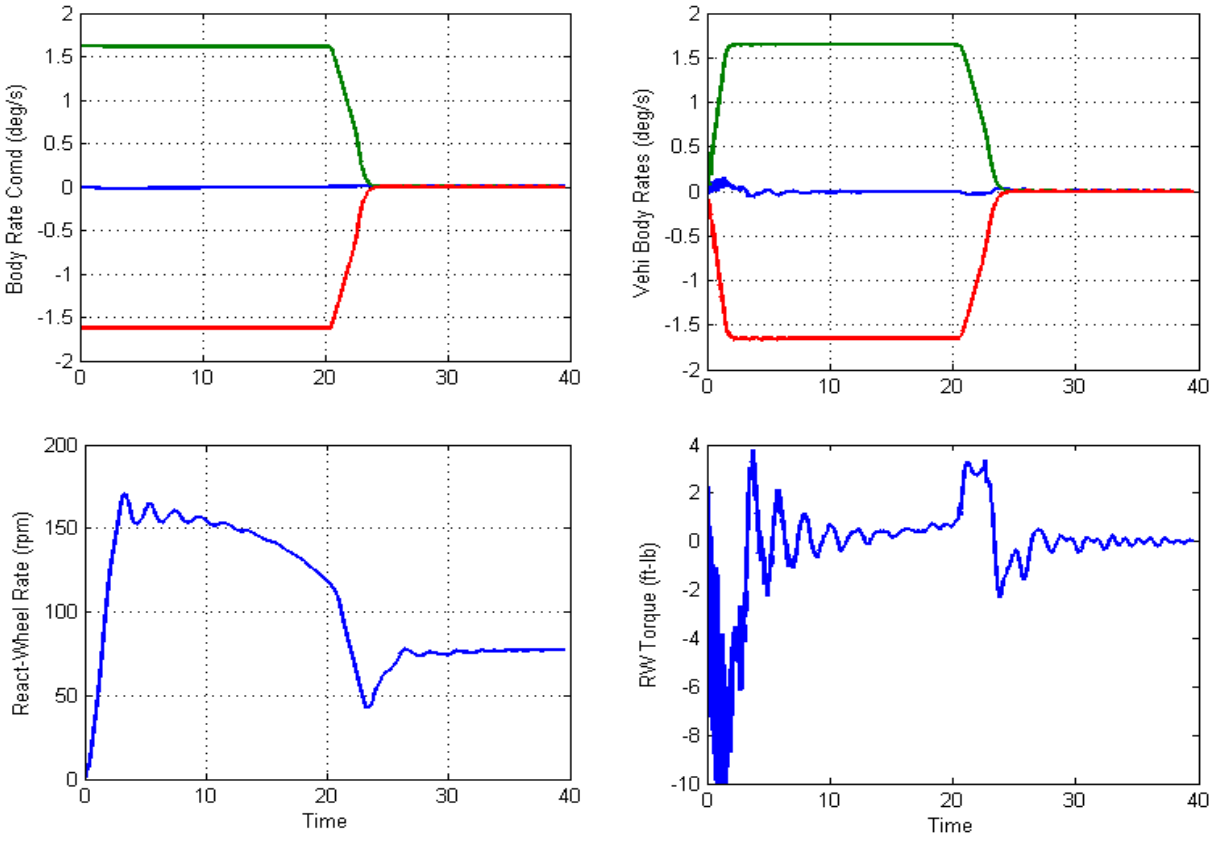
50 (deg) +Pitch and -Yaw Combination Maneuver: [0 1 -1]

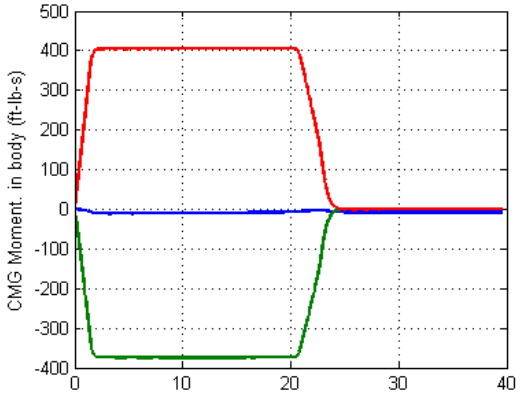
Let us begin with an easy maneuver. Maneuvers in diagonal pitch/ yaw directions perform great because the induced roll torques from the two SGCMGs cancel each other out. In the absence of roll disturbances the pitch and yaw CMG control system can perform ideal eigenaxis rotations, as shown below, for maneuvers which are even greater than 40º. There is some RW activity due to cross-coupling in roll because the system does not have perfect symmetry. Notice that, although the gimbal angles reach almost 40 (deg) the CMG gimbal torques are less than 34 (ft-lb) because the RW momentum is small and it does not create big (ω x h) torques against the gimbals.

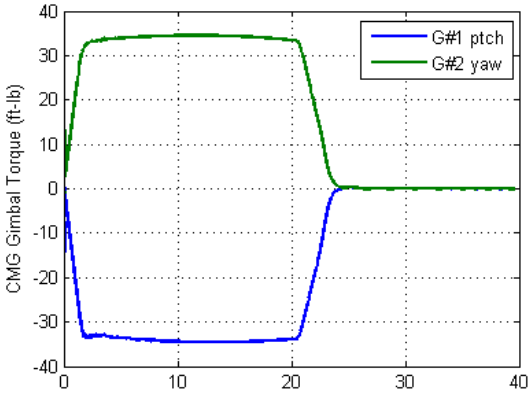


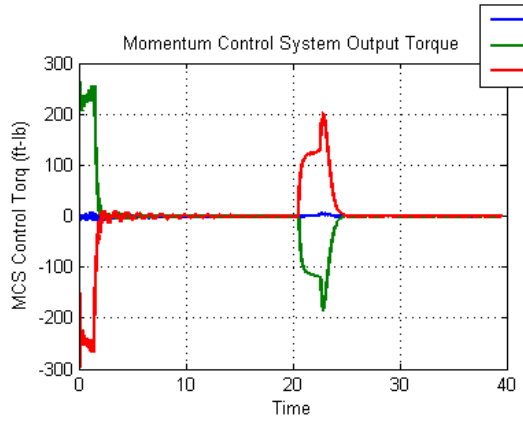


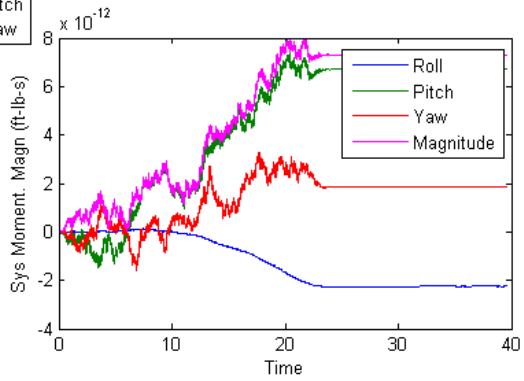
2 SGCMG & 1 RW with Flex, 60 deg maneuver in [0, 1, -1]



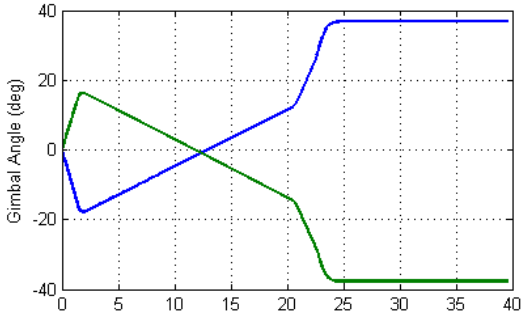


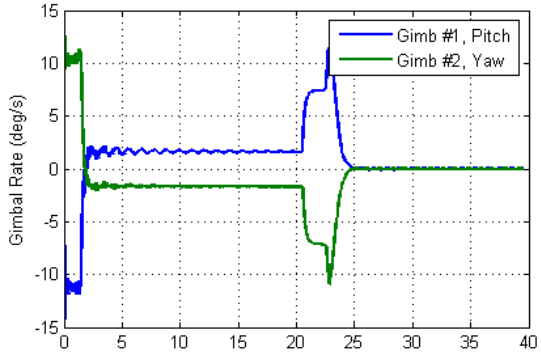


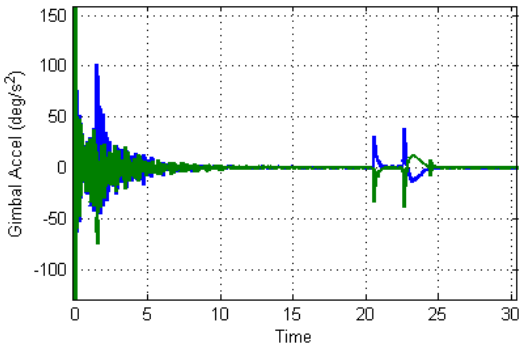


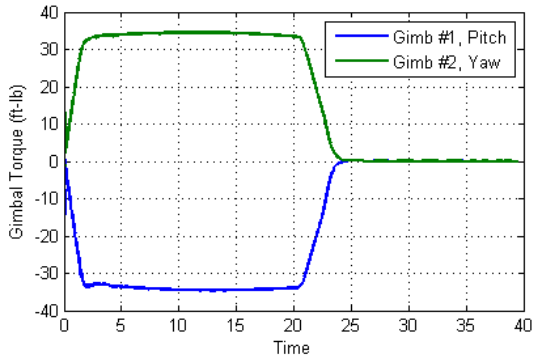


2 SGCMG & 1 RW with Flex, 60 deg maneuver in [0, 1, -1]



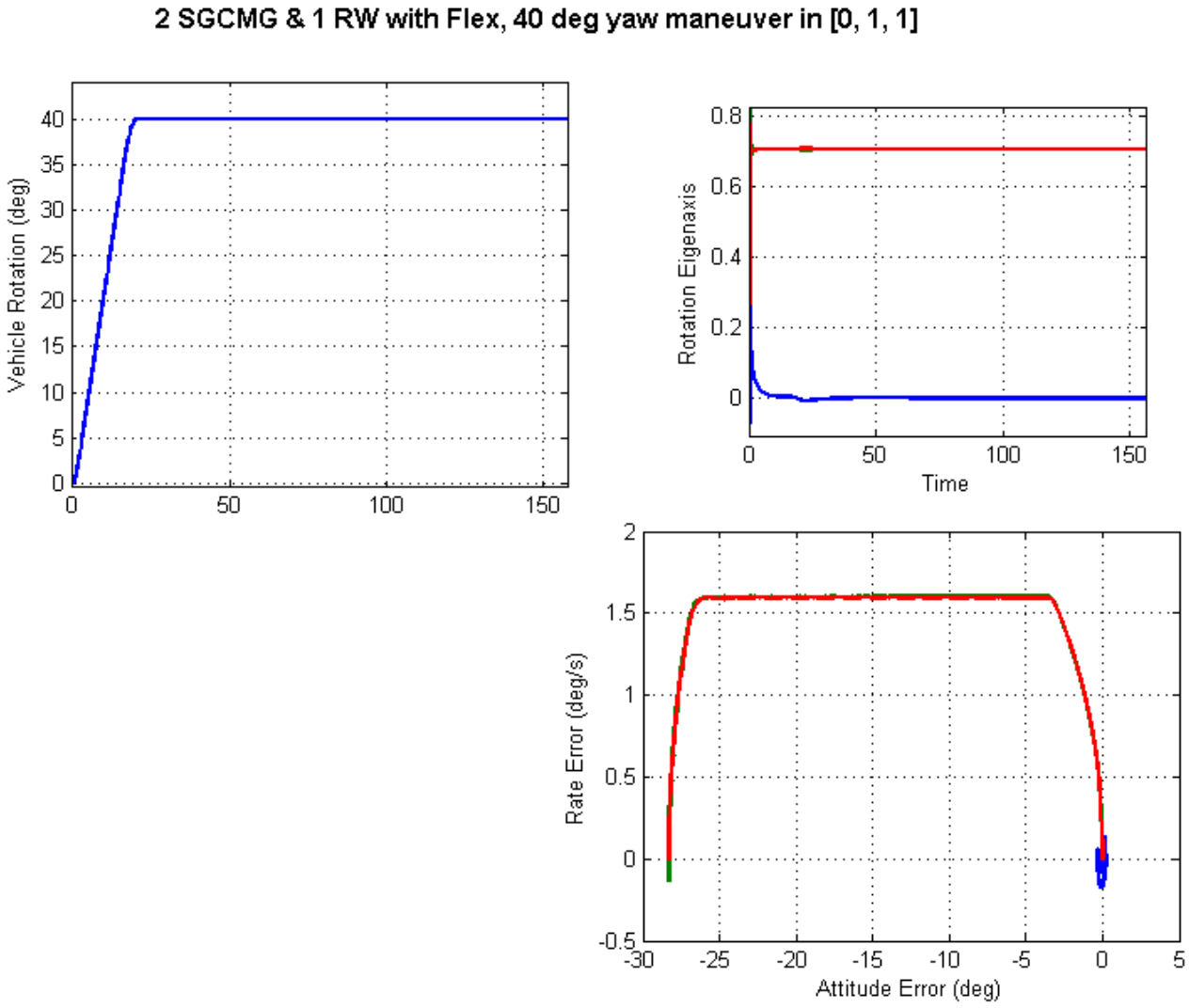


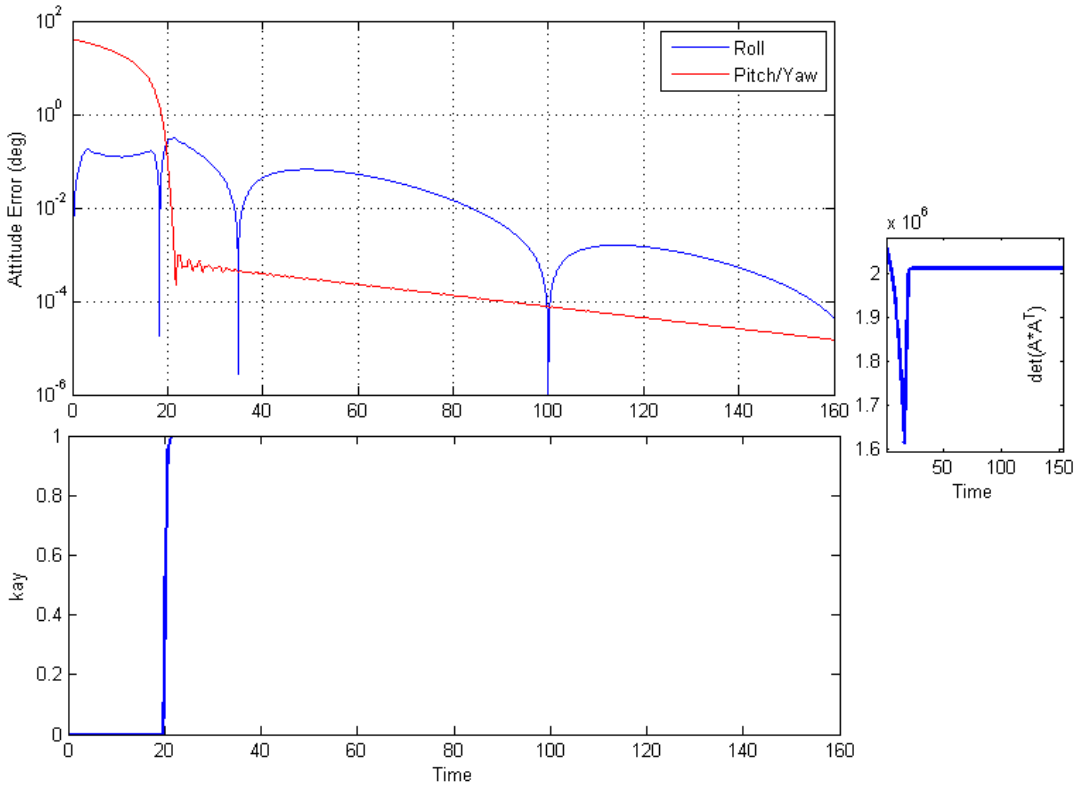




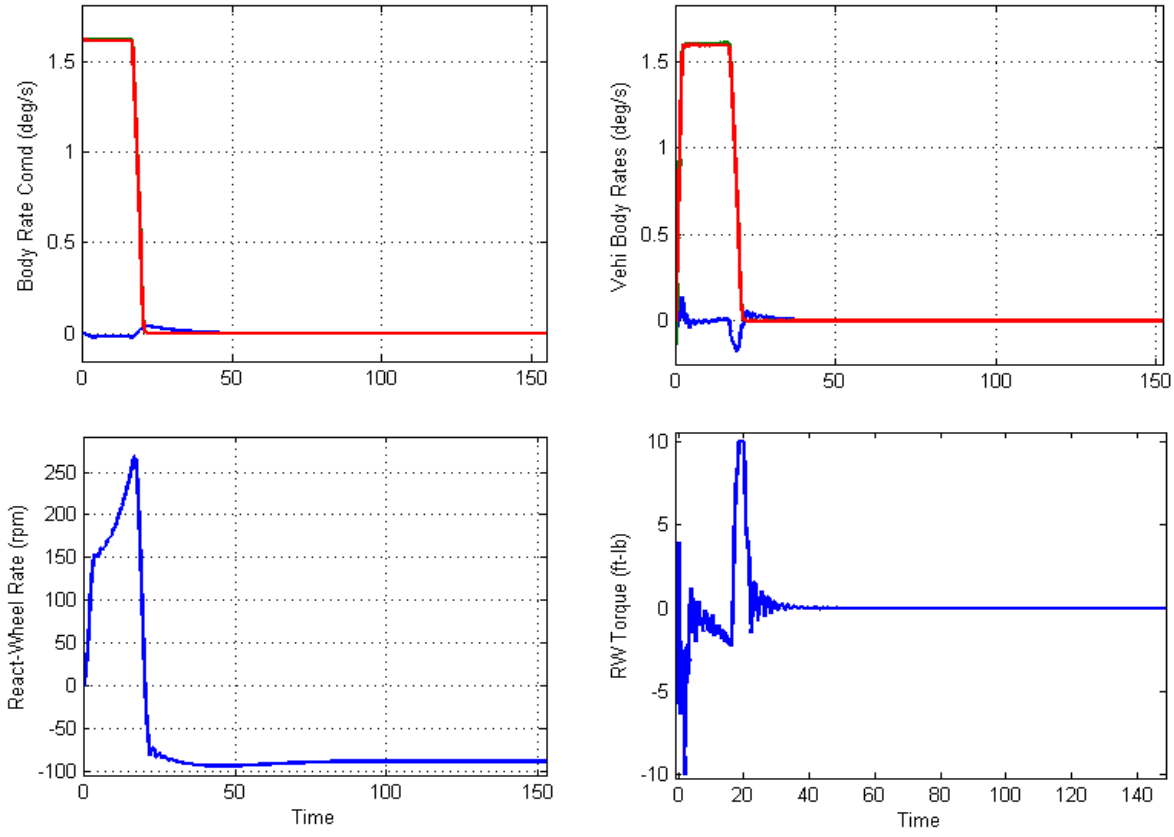
40 (deg) +Pitch and +Yaw Maneuver indirection : [0, 1, 1]

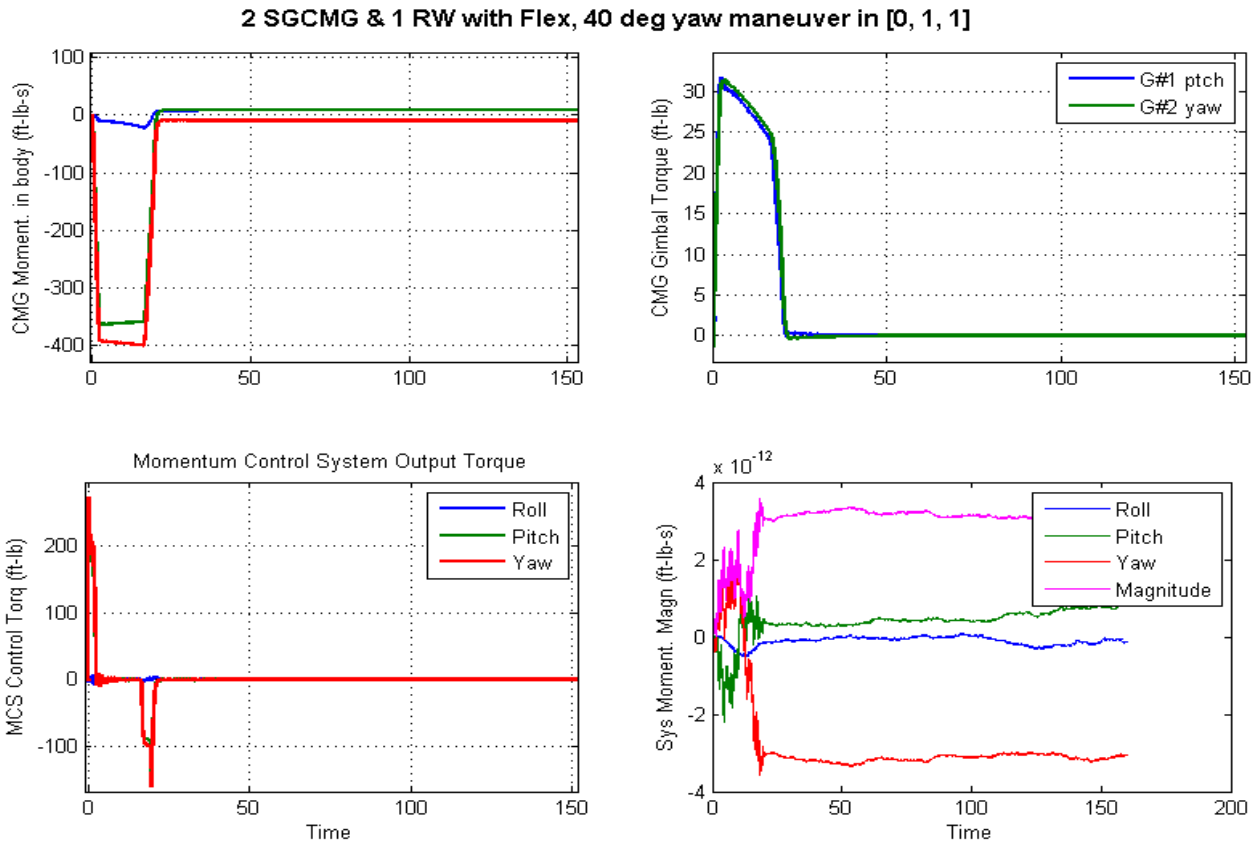
This one is another well behaved pitch/yaw 40º diagonal maneuver very similar to the previous one. It performs a perfect eigenaxis rotation with very little coupling in roll. Notice how the pitch/yaw error drops significantly when the integrators in the PID are turned on and the error continues to decay further. The roll error decays at a slower pace. Similar to the previous case, the RW rate reaches only 250 (rpm) and the CMG gimbal torques approx. 30 (ft-lb). The MCS control torque consists of both: CMG and RW torques. In this case it is mainly due to the CMGs. Notice also that, the body rates during phase-plane control are equal 1.7 º/sec in both pitch and yaw. The CMG momentum in pitch and yaw are negative since they are exchanging momentum with the spacecraft. Their sizes are not equal because the pitch and yaw moments of inertia are different. Notice also that, the plots should be labeled pitch/yaw instead of yaw.

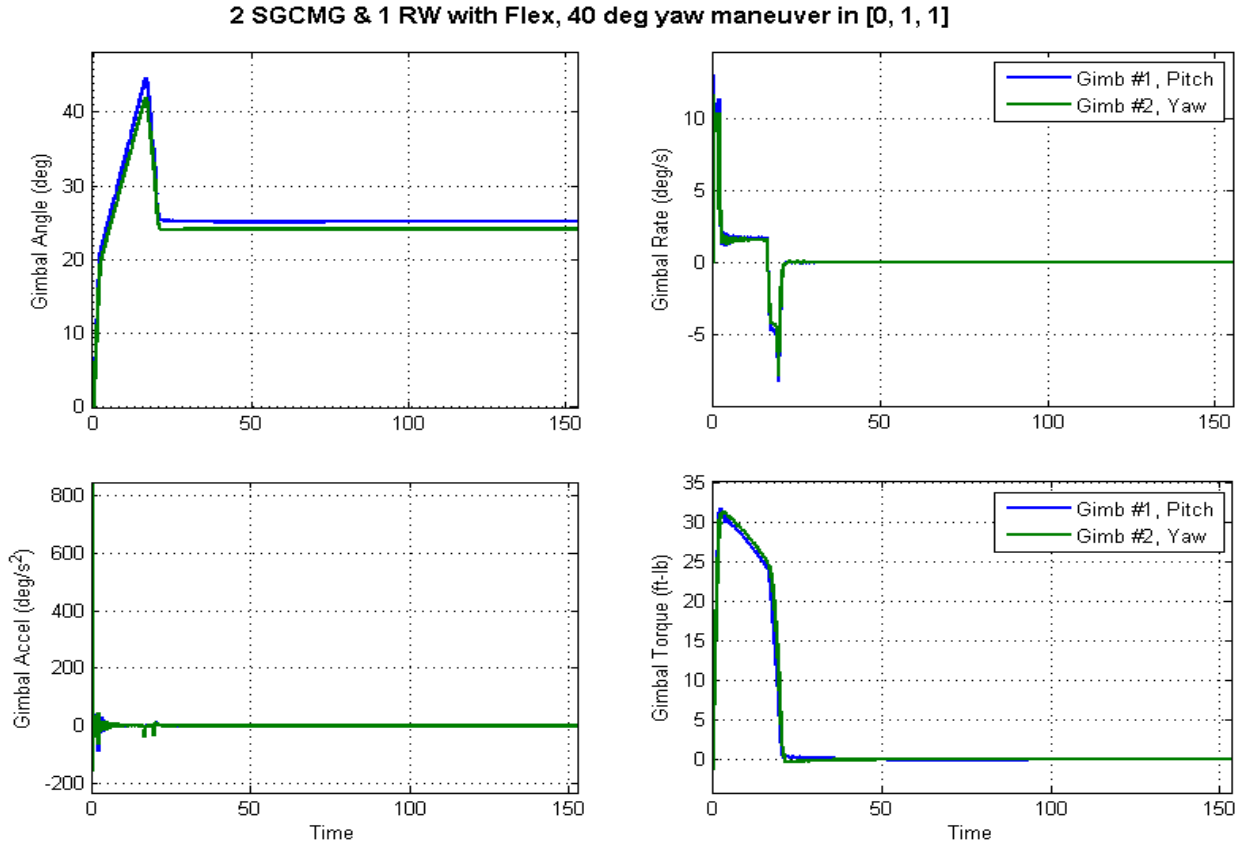




2 SGCMG & 1 RW with Flex, 40 deg yaw maneuver in [0, 1, 1]

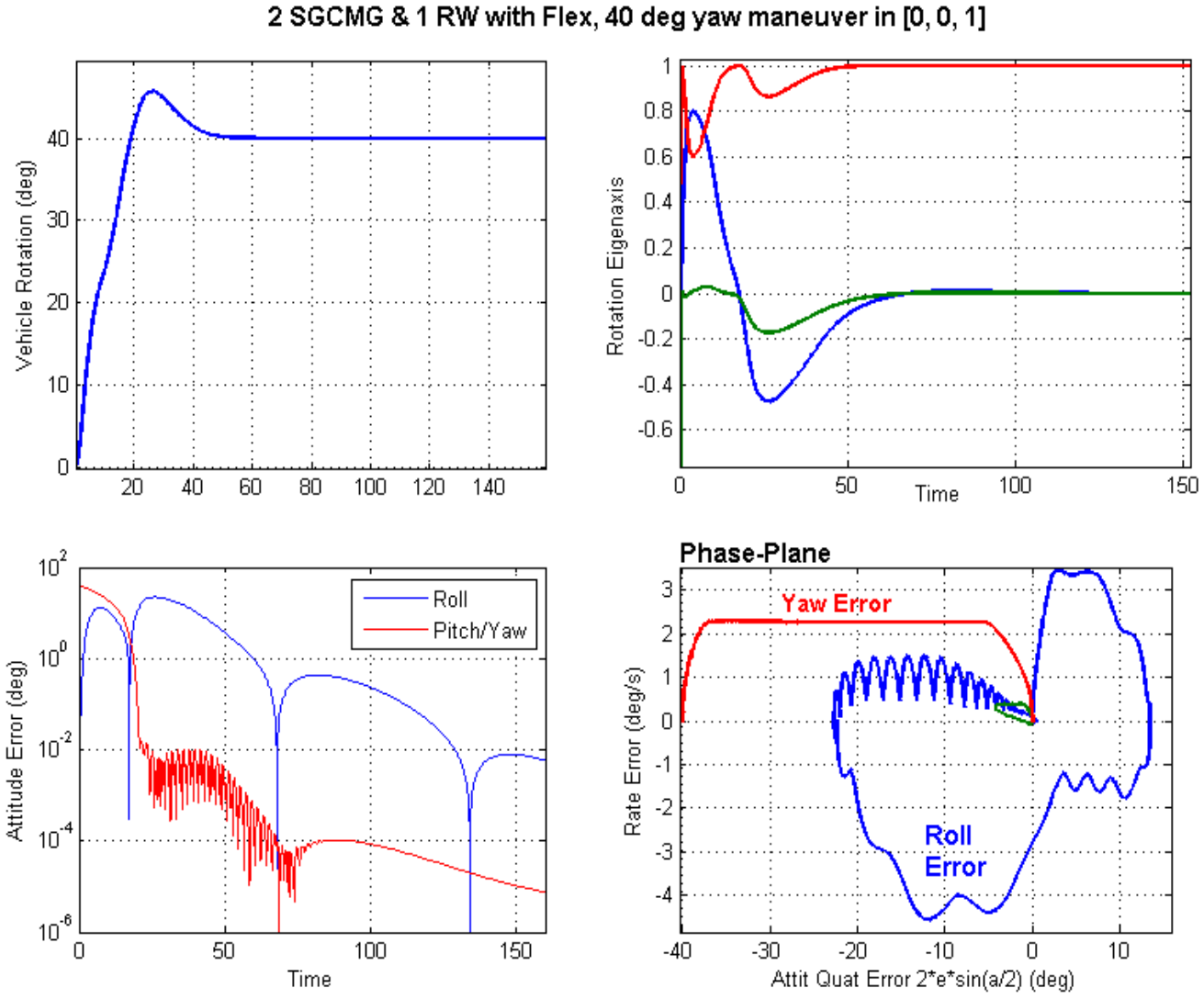


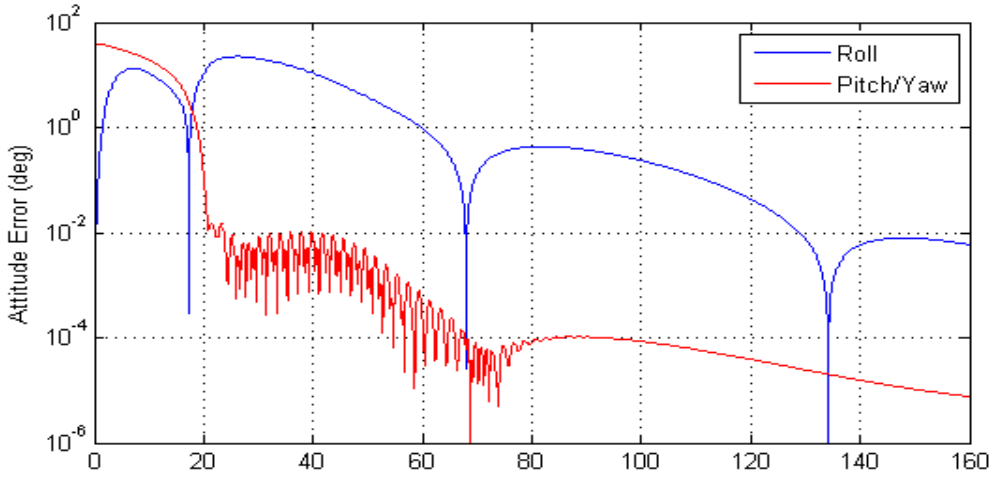

2 SGCMG & 1 RW with Flex, 40 deg yaw maneuver in [0, 1, 1]
CMG Moment. in body (ft-lb-s)
CMG Gimbal Torque (ft-lb)
G#1 ptch
G#2 yaw
Momentum Control System Output Torque
MCS Control Torq (ft-lb)
Roll
Pitch
Yaw
Time
x 10^-12
Sys Moment. Magn (ft-lb-s)
Roll
Pitch
Yaw
Magnitude
Time

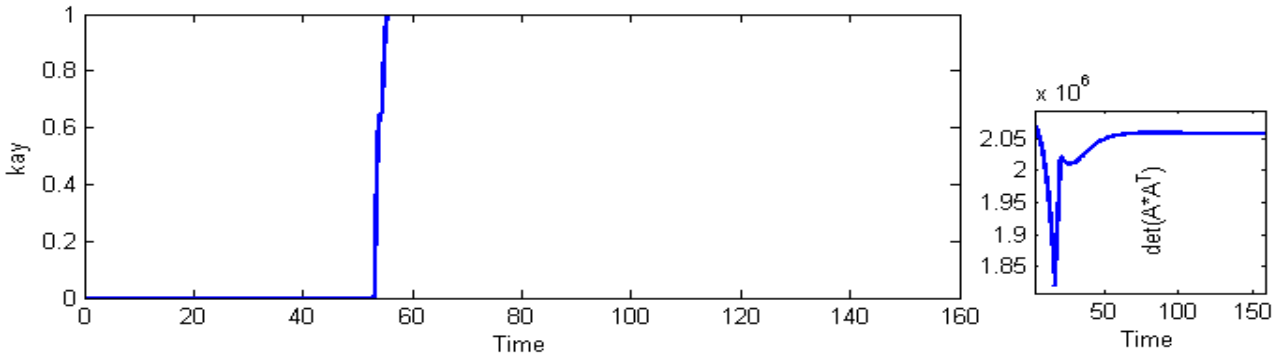

2 SGCMG & 1 RW with Flex, 40 deg yaw maneuver in [0, 1, 1]
Gimbal Angle (deg)
Gimbal Rate (deg/s)
Gimb #1, Pitch
Gimb #2, Yaw
Gimbal Accel (deg/s^2)
Gimbal Torque (ft-lb)
Gimb #1, Pitch
Gimb #2, Yaw
Time
Time

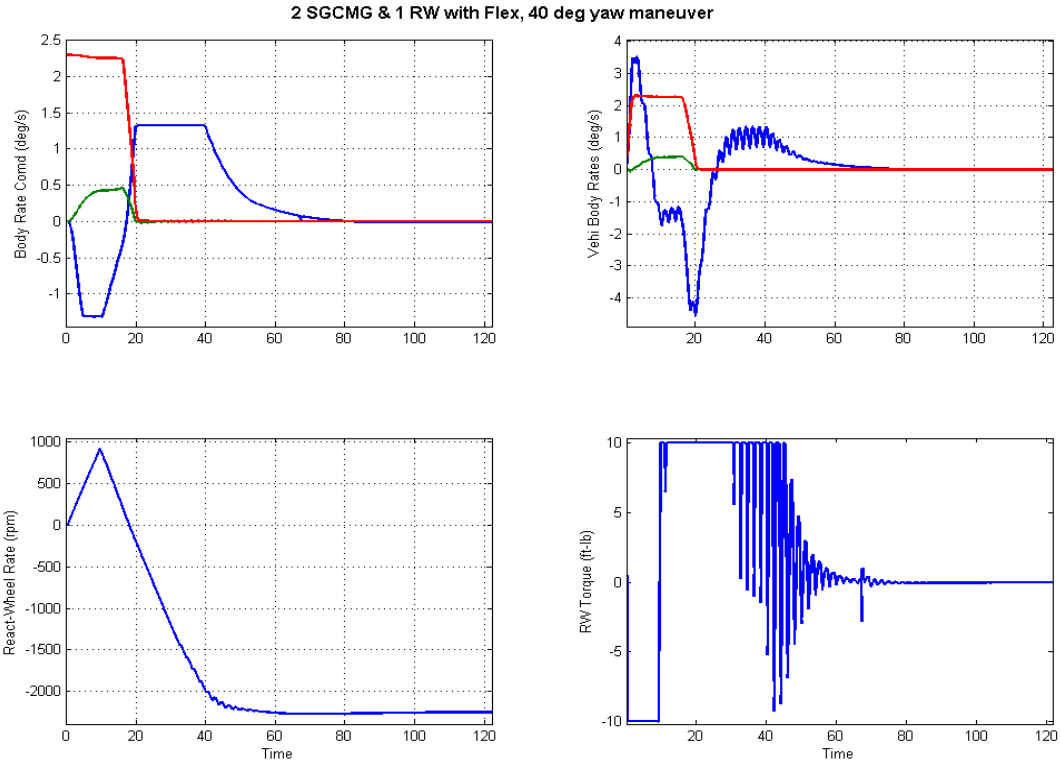
40 (deg) Yaw Maneuver

The following maneuver does not look as great as the previous two although it meets the requirements 100%. The vehicle is commanded to perform a 40º rotation in yaw. It executes the rotation while it induces a big transient in roll, as shown in the phase-plane. The yaw maneuver requires a pitch CMG gimbal rate action and an 80 (ft-lb) pitch gimbal torque. The CMG induces a significant amount of roll disturbance that the RW tries to compensate against it but it saturates due to its limited torque. This causes a RW torque chattering that couples slightly in the yaw gimbal rate and the pitch CMG torque.

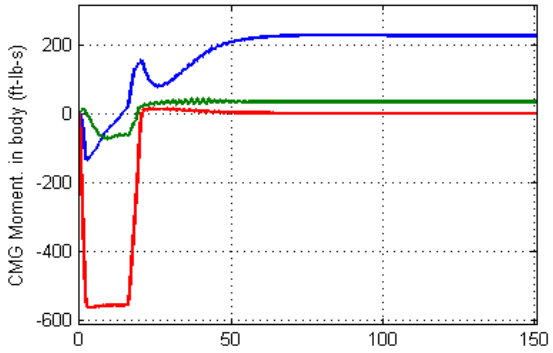


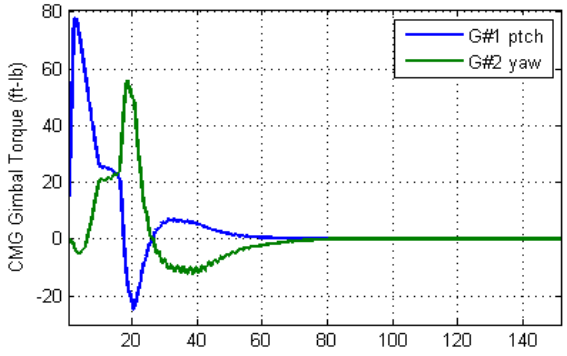

Roll
Pitch/Yaw
Attitude Error (deg)

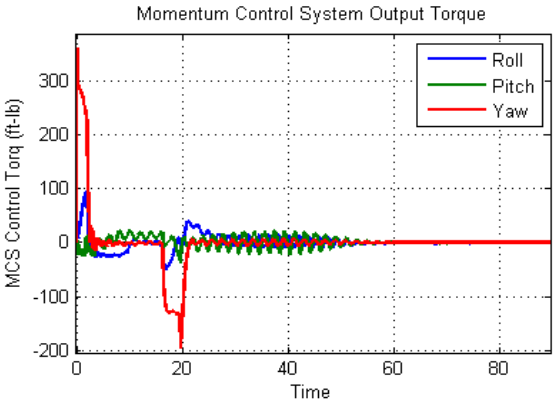

kay
Time
x 10^6
det(A*A^T)
Time

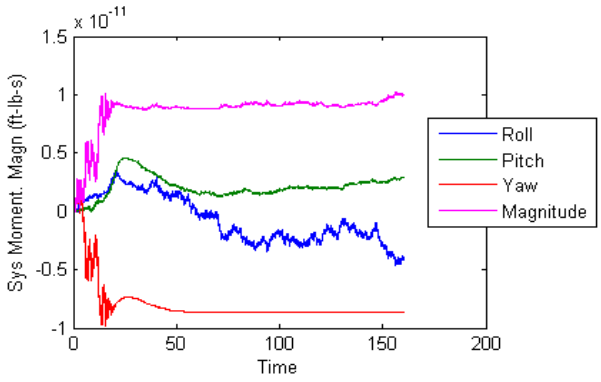

2 SGCMG & 1 RW with Flex, 40 deg yaw maneuver
Body Rate Comd (deg/s)
Vehi Body Rates (deg/s)
React-Wheel Rate (rpm)
Time
RW Torque (ft-lb)
Time

2 SGCMG & 1 RW with Flex, 40 deg yaw maneuver in [0, 0, 1]

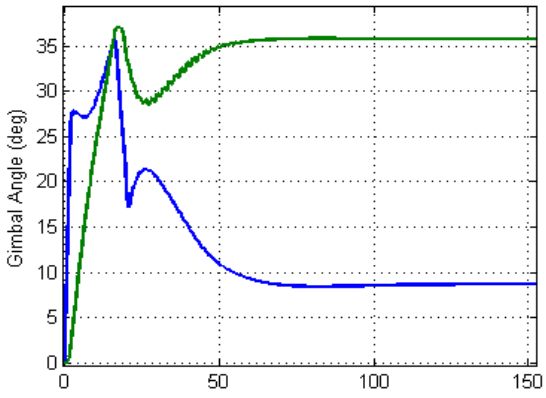


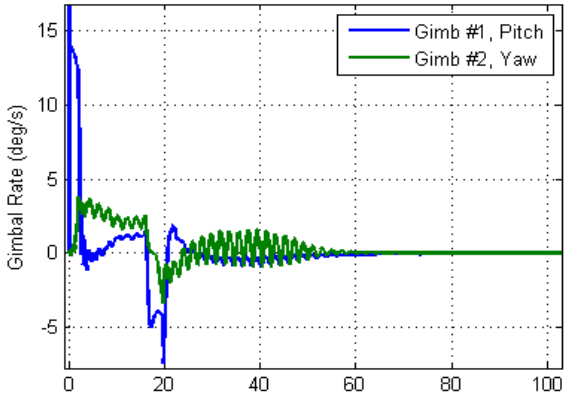


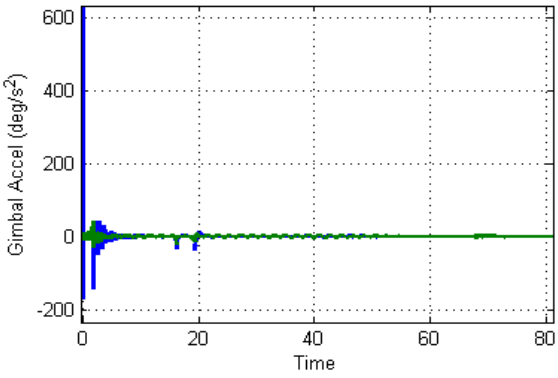


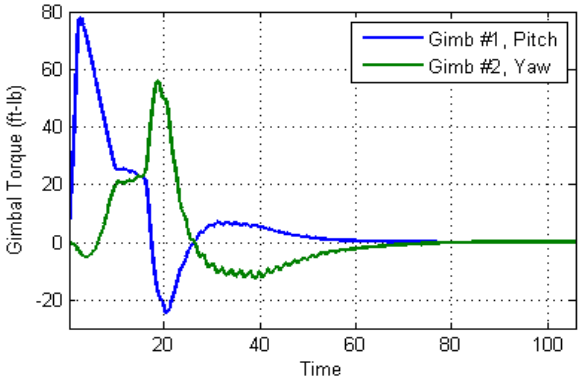


2 SGCMG & 1 RW with Flex, 40 deg yaw maneuver in [0, 0, 1]



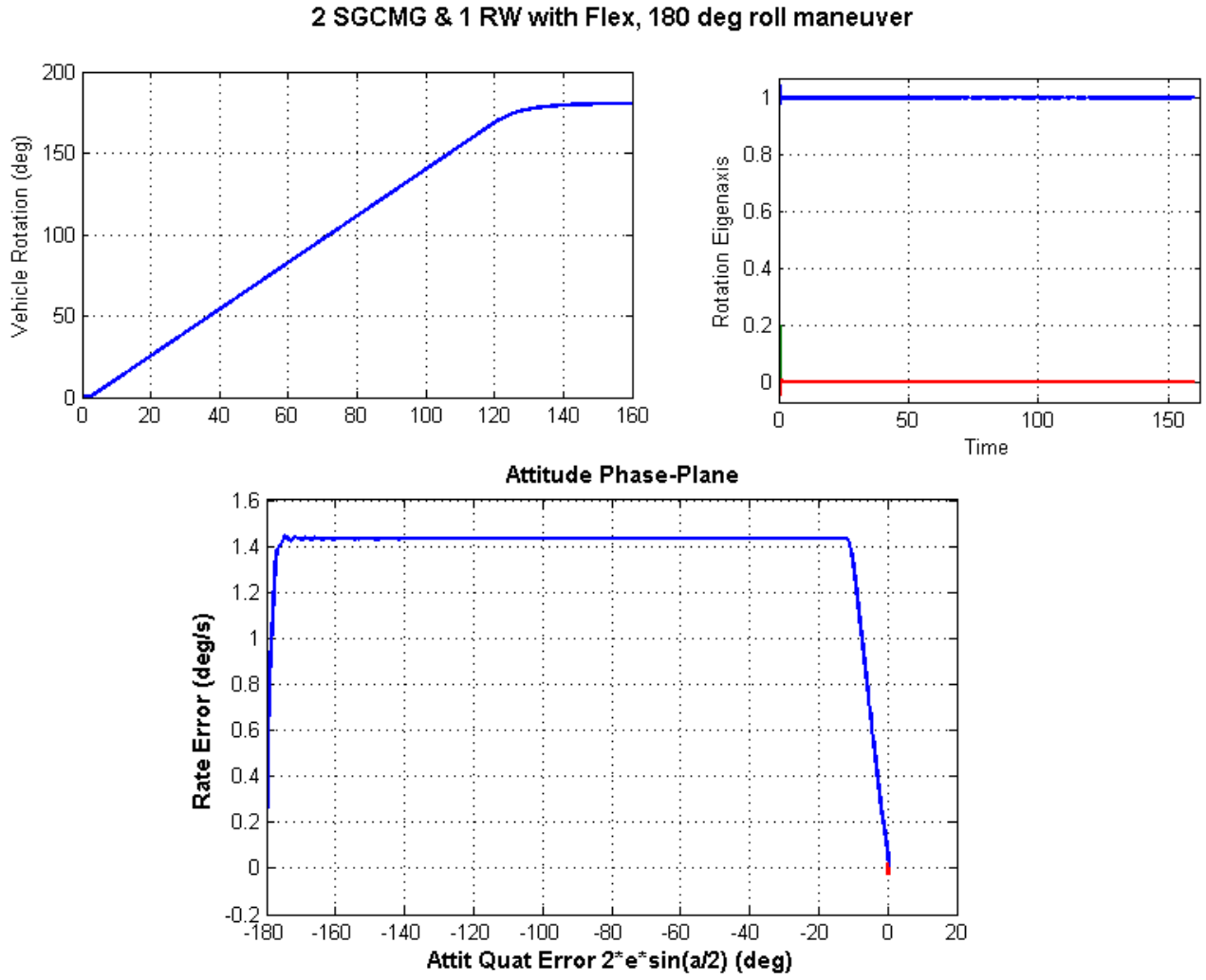


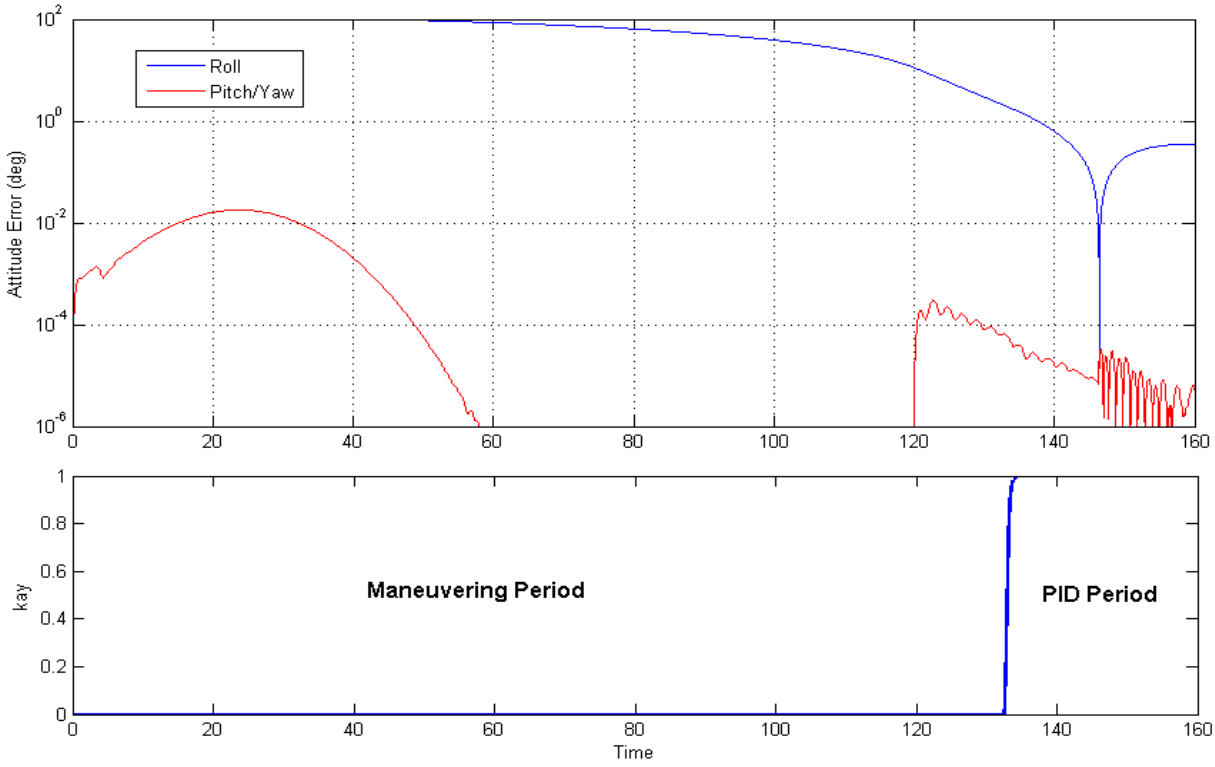




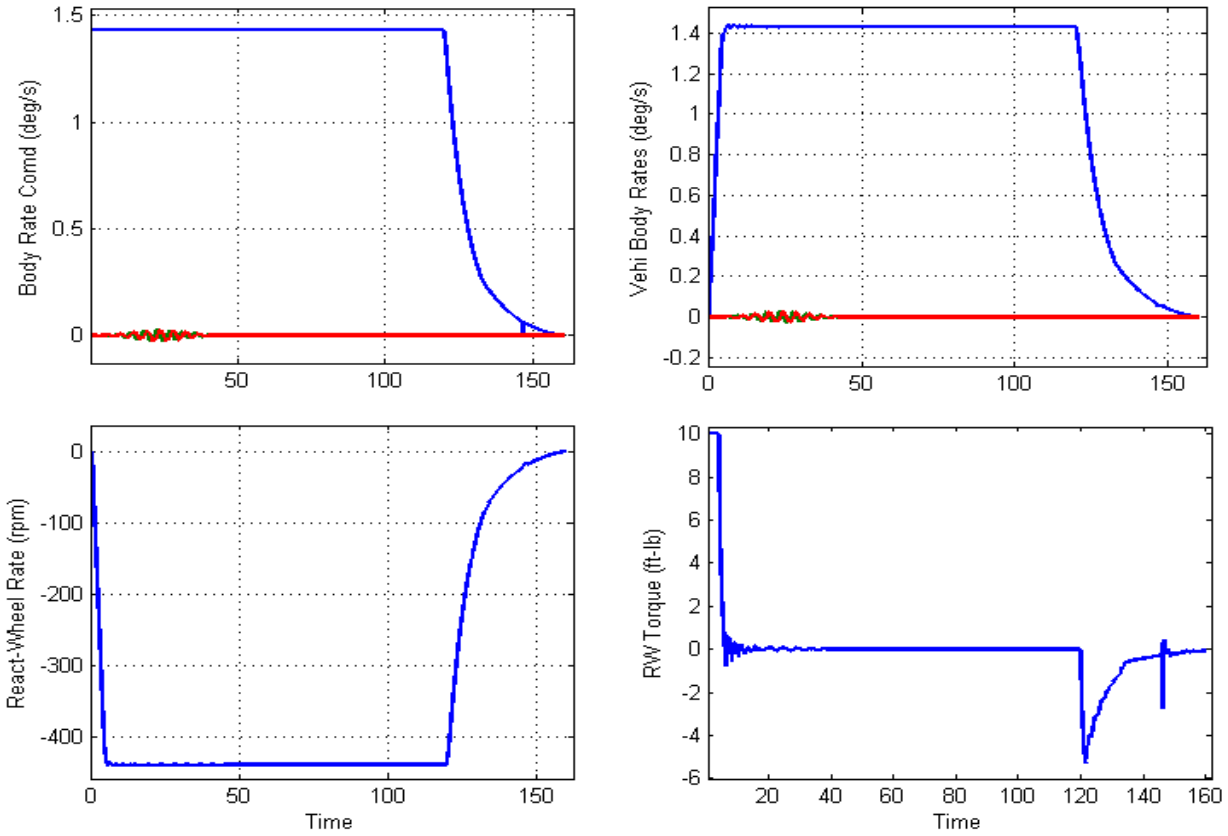
180 (deg) Roll Maneuver

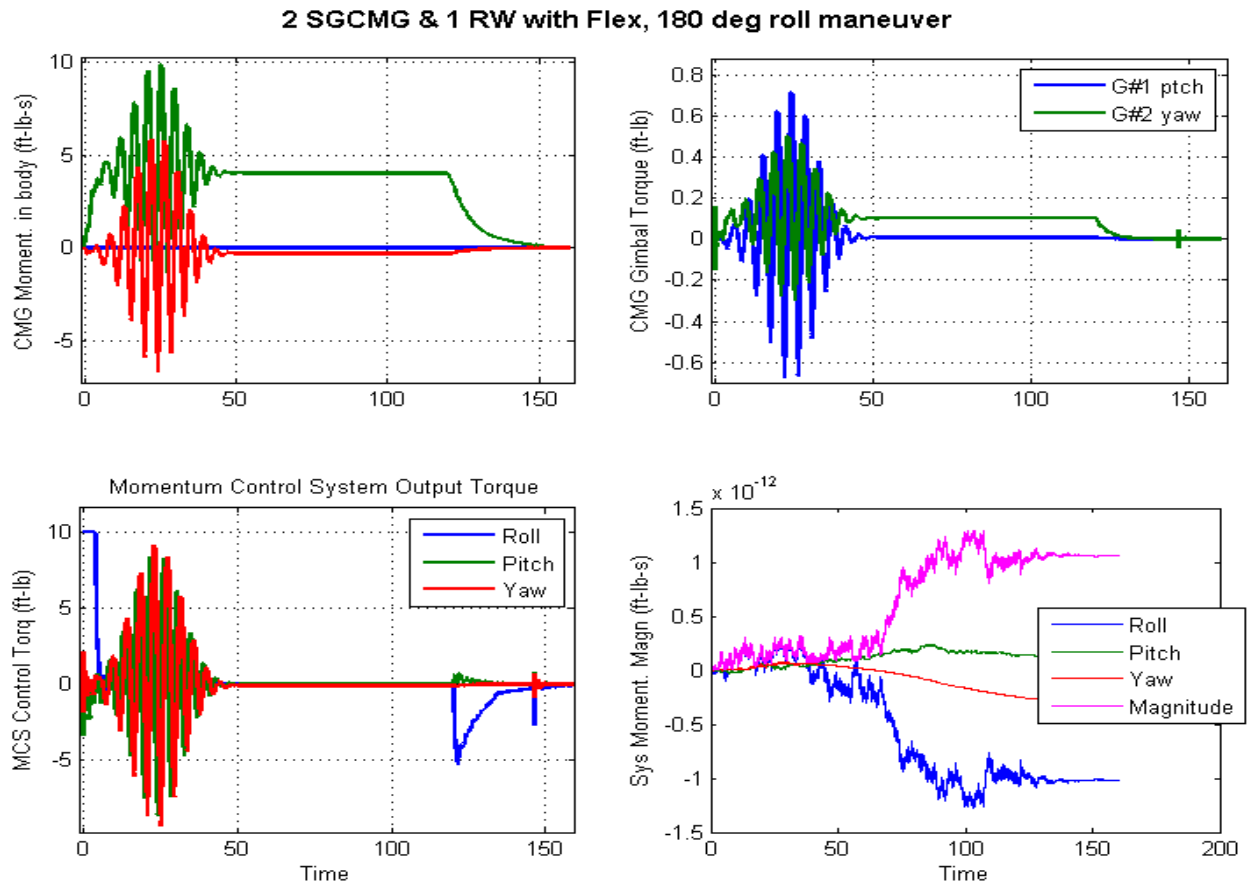
Although the spacecraft typically maintains zero roll attitude while it is maneuvering in pitch and yaw, it still has the capability to perform roll maneuvers when needed. In the following case the spacecraft uses mainly the reaction wheel to rotate 180º about the line-of-sight (roll) axis. The command direction vector is [1, 0, 0] and the rotation angle is 180º. The eigenaxis plot below shows that the rotation is only in roll without any coupling in the other two axes. The phase-plane shows that the spacecraft roll rate reaches a max steady value of 1.42 (deg/sec) and maintains that rate during most of the phase-plane portion of the maneuver. At the same time the RW reaches a max rate of 430 (rpm). The integrators are turned on when the roll error drops below 1º and the PID controller helps to further attenuate the attitude error. The maneuver causes a small transient due to cross coupling at the CMG gimbal rates that eventually dies down.

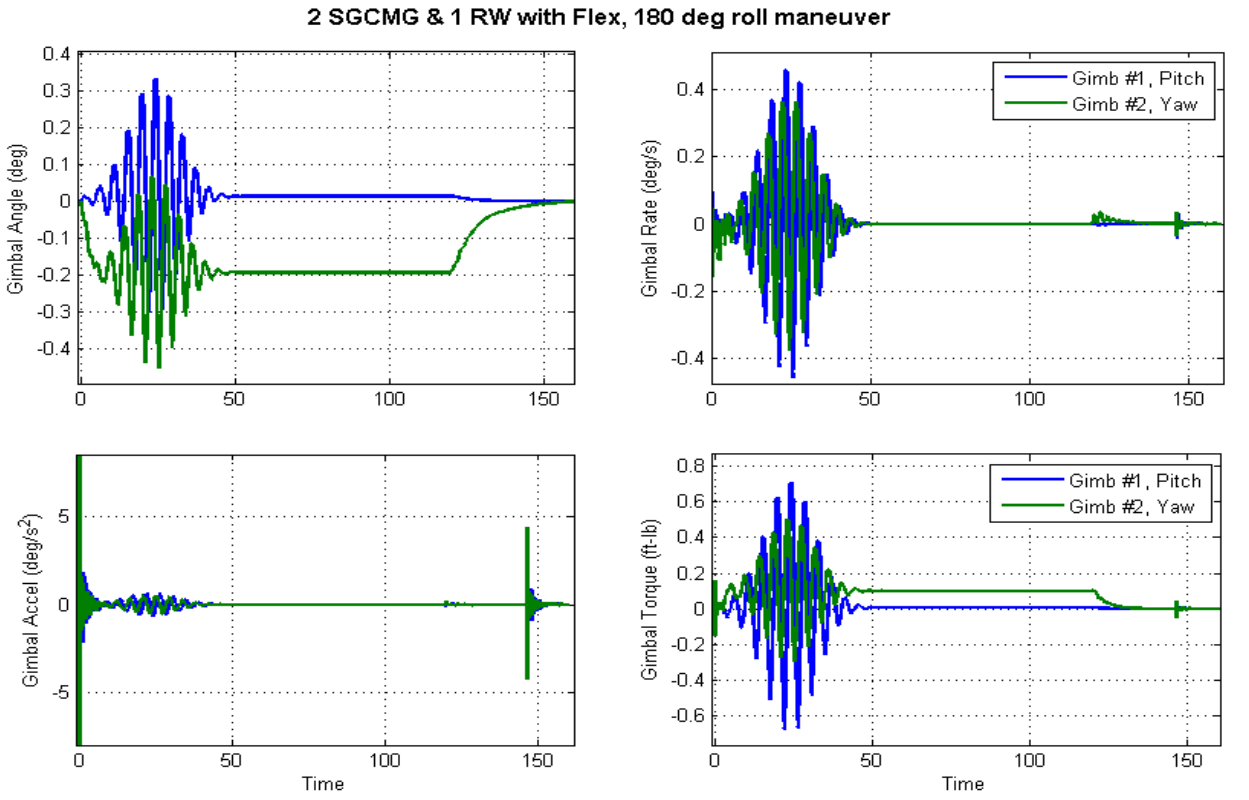




2 SGCMG & 1 RW with Flex, 180 deg roll maneuver

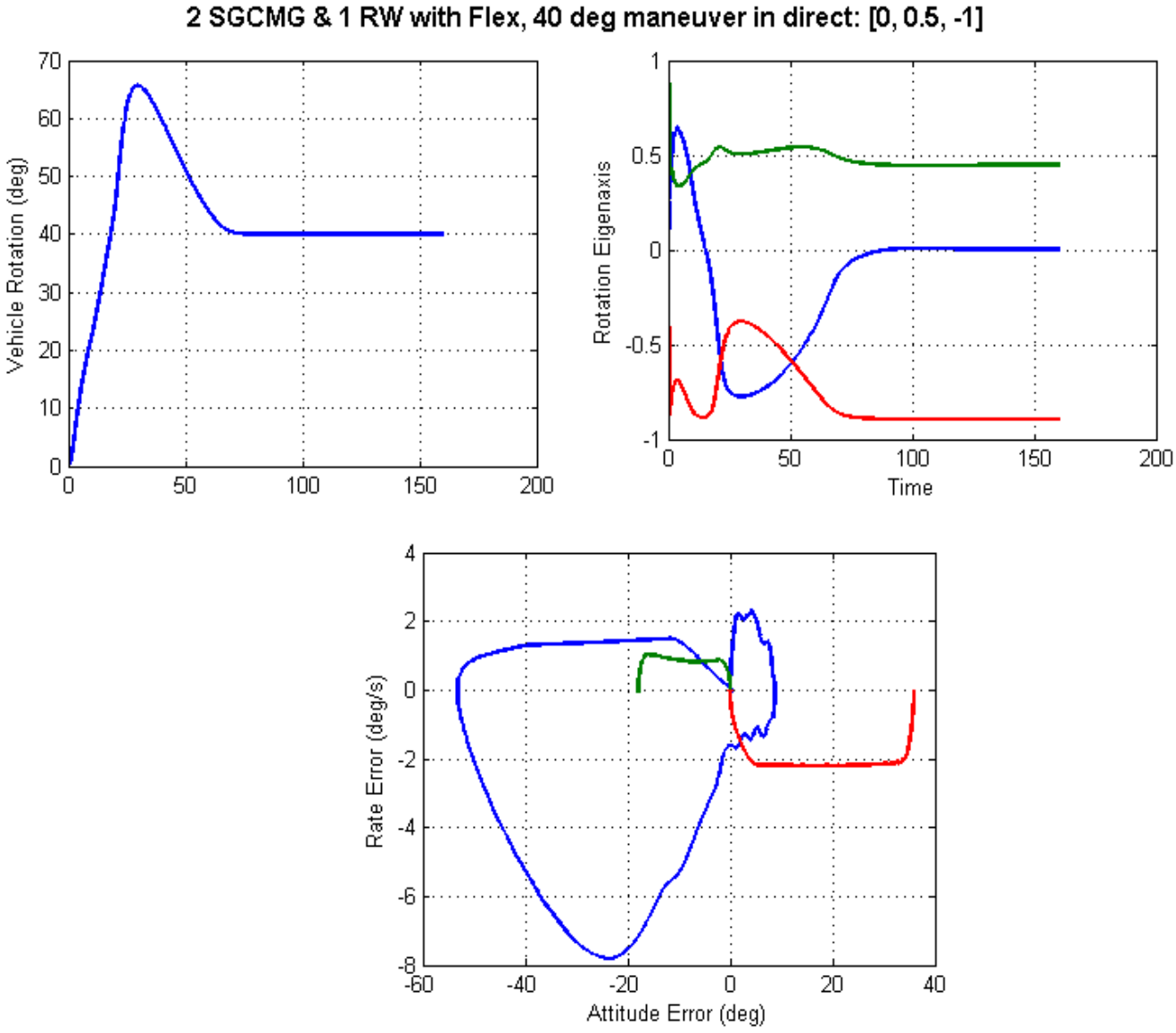


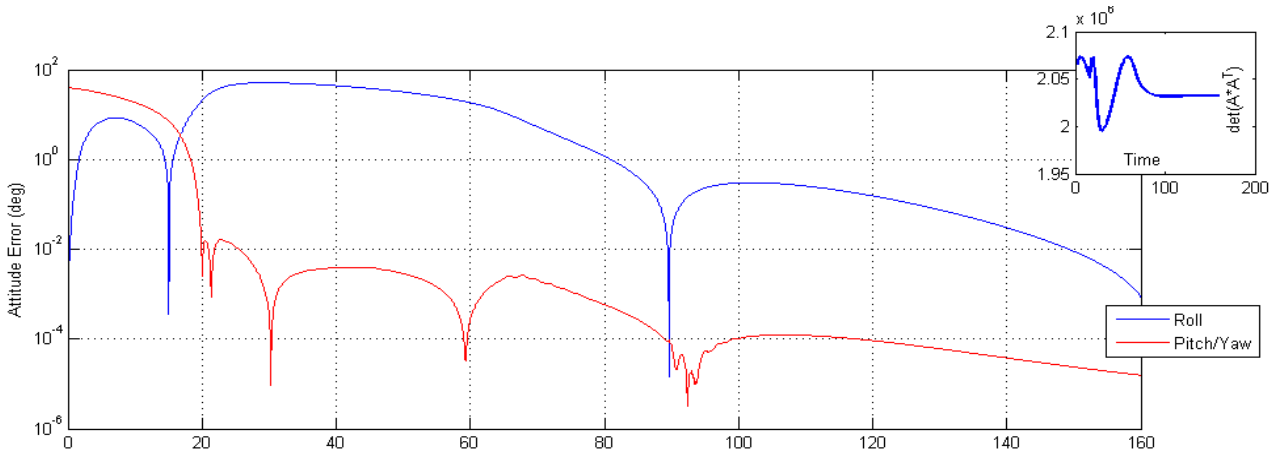

2 SGCMG & 1 RW with Flex, 180 deg roll maneuver
CMG Moment. in body (ft-lb-s)
CMG Gimbal Torque (ft-lb)
G#1 ptch
G#2 yaw
Momentum Control System Output Torque
MCS Control Torq (ft-lb)
Roll
Pitch
Yaw
Time
x 10^-12
Sys Moment. Magn (ft-lb-s)
Roll
Pitch
Yaw
Magnitude
Time

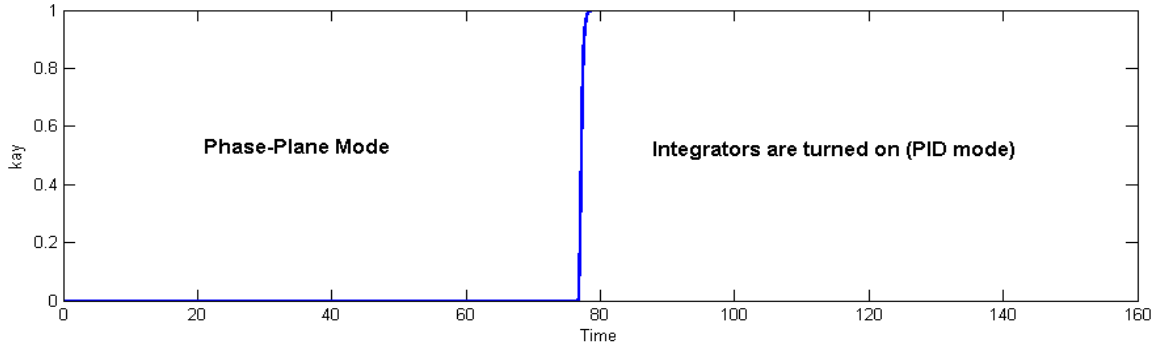

2 SGCMG & 1 RW with Flex, 180 deg roll maneuver
Gimbal Angle (deg)
Gimbal Rate (deg/s)
Gimb #1, Pitch
Gimb #2, Yaw
Gimbal Accel (deg/s^2)
Gimbal Torque (ft-lb)
Gimb #1, Pitch
Gimb #2, Yaw
Time
Time

40 (deg) Maneuver in direction: [0, 0.5 -1]

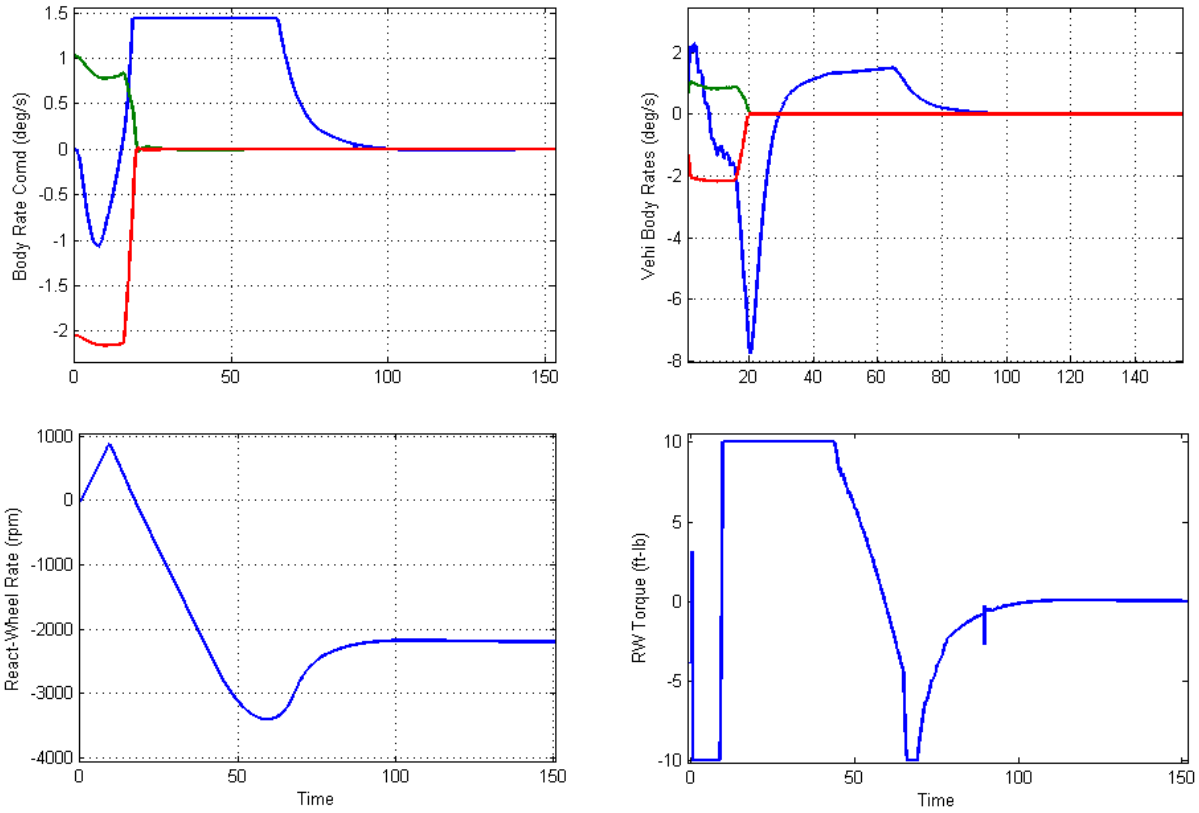
This maneuver represents one of the bad cases, although it still meets the pointing requirement. It is a pitch and mostly -yaw combination maneuver that creates a significant amount of coupling in the roll axis, and since the roll axis is slow by design it takes longer to converge. The CMGs which are primarily used in pitch and yaw generate in this case a significant amount of roll disturbance (see how the vehicle roll rate deviates from the commanded rate). This makes the reaction wheel torque to saturate resulting to a huge roll error and it takes longer to converge. The pitch and yaw errors, however, which are of more importance than roll, are attenuated pretty significantly, while the roll error is also attenuated but at a slower rate. The rate of error decay improves when the ACS switches to PID mode.

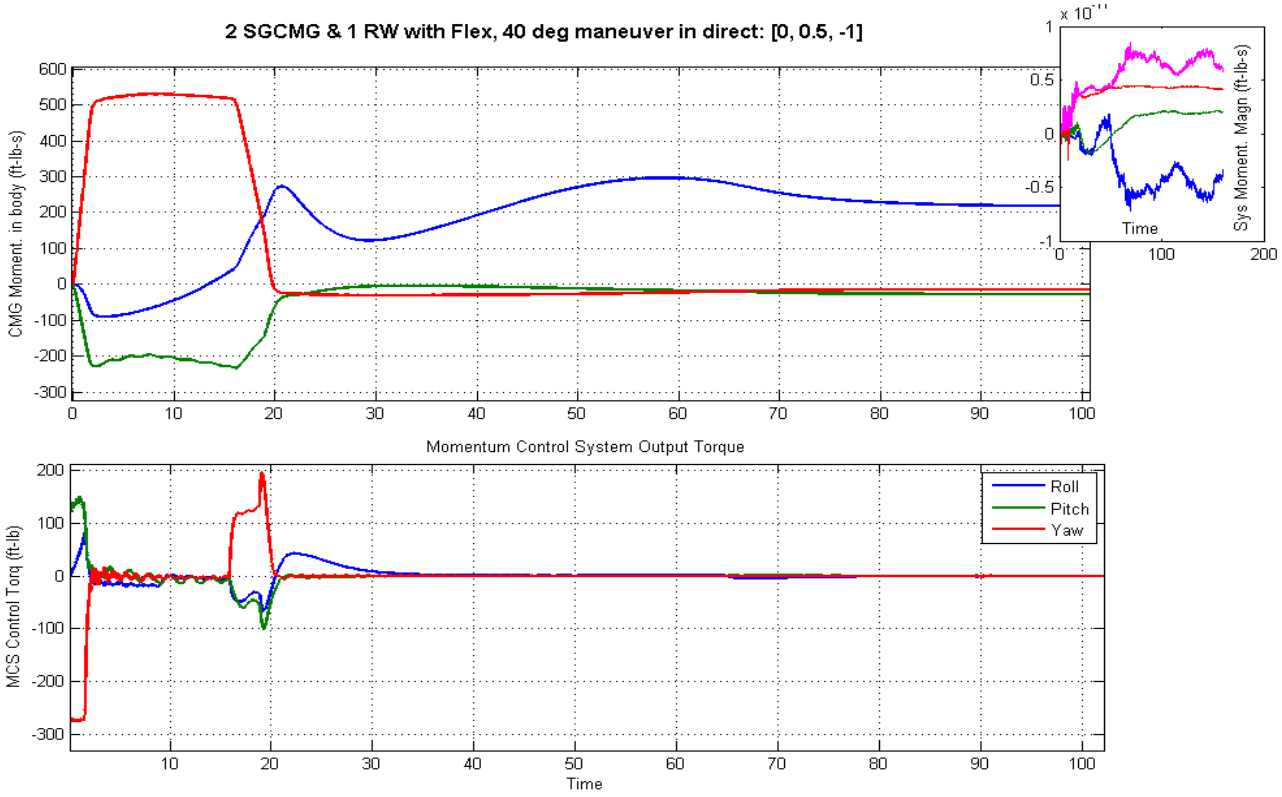


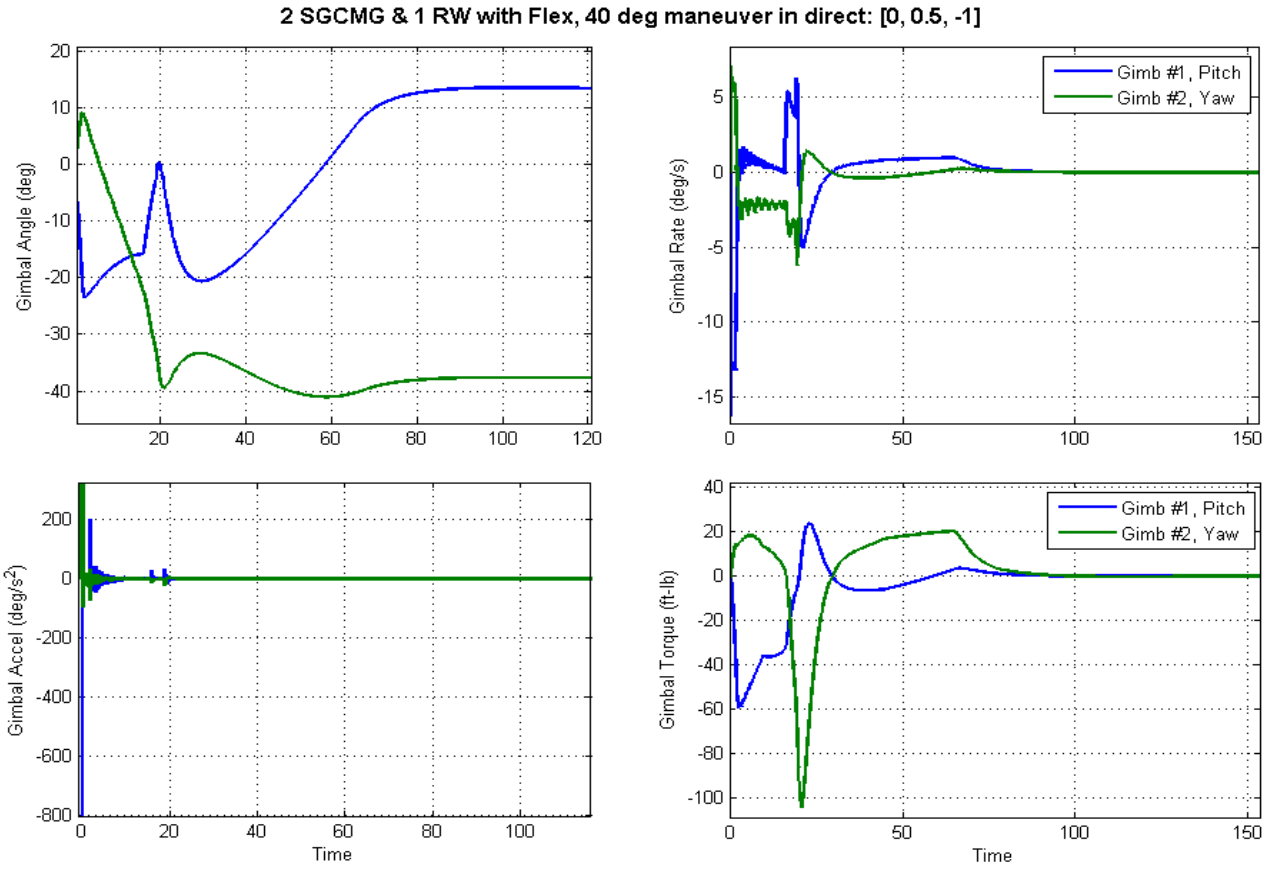




2 SGCMG & 1 RW with Flex, 40 deg maneuver in direct: [0, 0.5, -1]




2 SGCMG & 1 RW with Flex, 40 deg maneuver in direct: [0, 0.5, -1]
CMG Moment. in body (ft-lb-s)
x 10^-11
Sys Moment. Magn (ft-lb-s)
Time
Momentum Control System Output Torque
MCS Control Torq (ft-lb)
Roll
Pitch
Yaw
Time


2 SGCMG & 1 RW with Flex, 40 deg maneuver in direct: [0, 0.5, -1]
Gimbal Angle (deg)
Gimbal Rate (deg/s)
Gimb #1, Pitch
Gimb #2, Yaw
Gimbal Accel (deg/s^2)
Gimbal Torque (ft-lb)
Time

Linear Stability Analysis

The Simulink model "*Open_Loop_Flex.mdl*", shown in Figure (9), is used for linear stability analysis. It is similar to the simulation model "*MaxEn_RW_2SGCMG_Flex.mdl*" but it is used for frequency response analysis and not simulations. It allows the loops to be opened at the spacecraft input torques. The loop is either opened at the RW torque command for frequency response calculation or at one of the two CMG gimbal torque inputs, while the other loops remain closed. Figure (3.6.11) below shows the loop opened at the CMG #2 yaw gimbal. The non-linearities have been taken out and this system represents the steady-state PID mode of operation. The attitude control system function was replaced with a linear ACS version using Matlab function "*ACS_Lin.m*" instead of "*MaxEn_ACS.m*". Also the steering inner loop controller was replaced with a linearized version Matlab function " *Steering_Lin.m*" instead of "*Steering_Lin.m*". The ACS is commanded at zero quaternion attitude. The Matlab file "frequ.m" is used to calculate the frequency response of the open-loop model in order to measure the phase and gain margins, as shown in the Bode and Nichols plots, Figures (3.6.12 to 14).

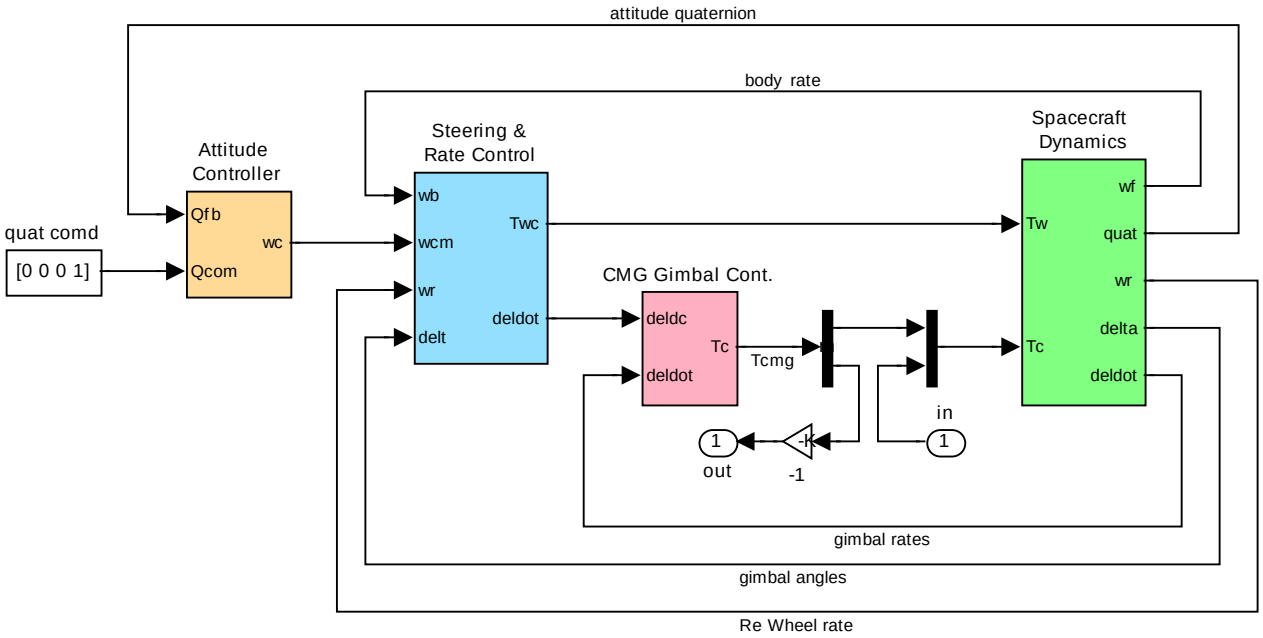


Figure 3.6.11 Simulink model " *Open_Loop_Flex.mdl* " used for open-loop frequency domain analysis

```
% Linear Analysis script freq.m
[Ao,Bo,Co,Do]=linmod('Open_Loop_Flex');         % Open-Loop Analysis Model
syso=SS(Ao,Bo,Co,Do);                           % for Stability analysis
w=logspace(-2,4,20000);
figure(2); Bode(syso,w); grid on
figure(3); Nichols(syso,w)
```

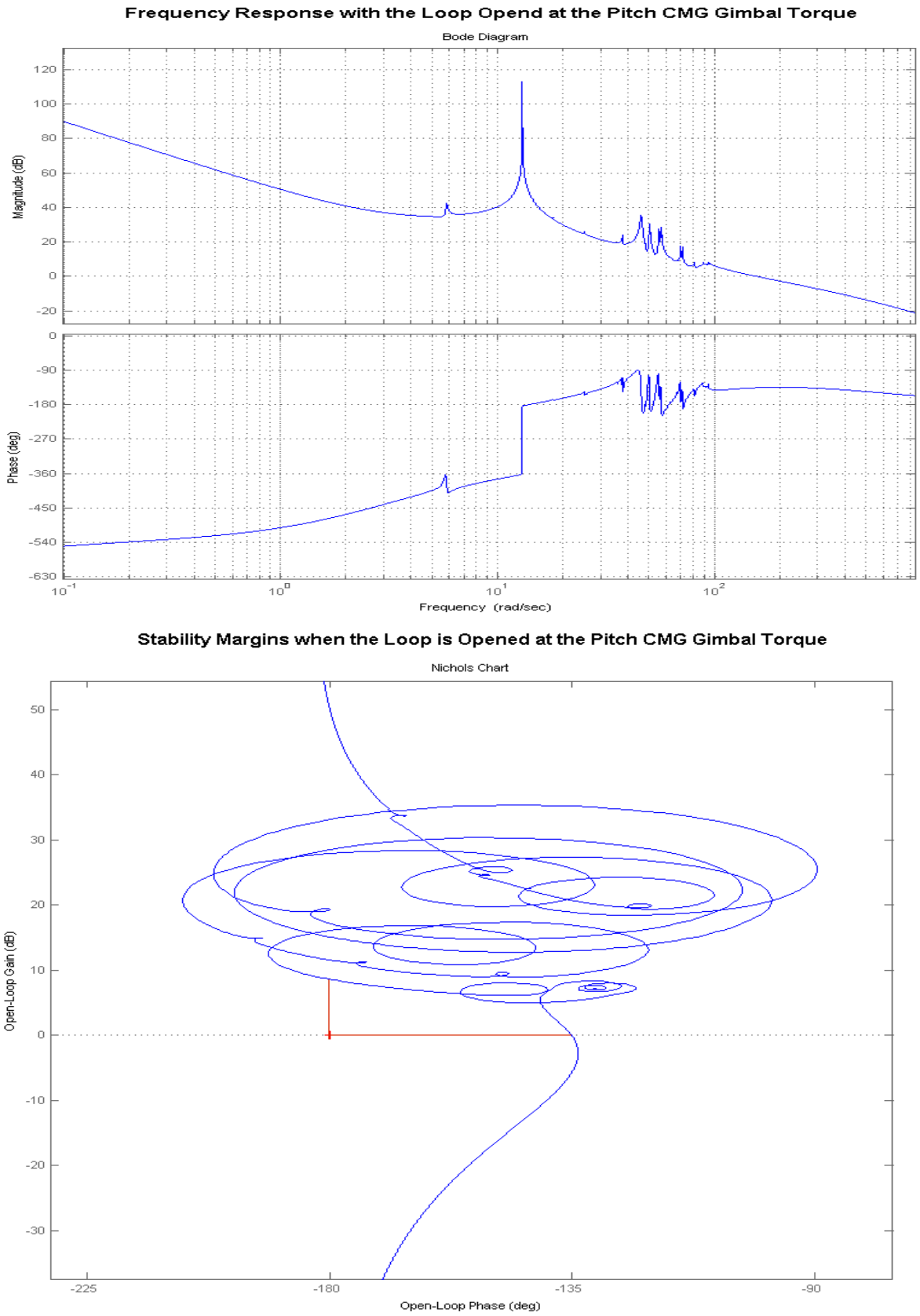


Figure 3.6.12 Frequency response with the loop opened at the pitch CMG gimbal showing stability margins

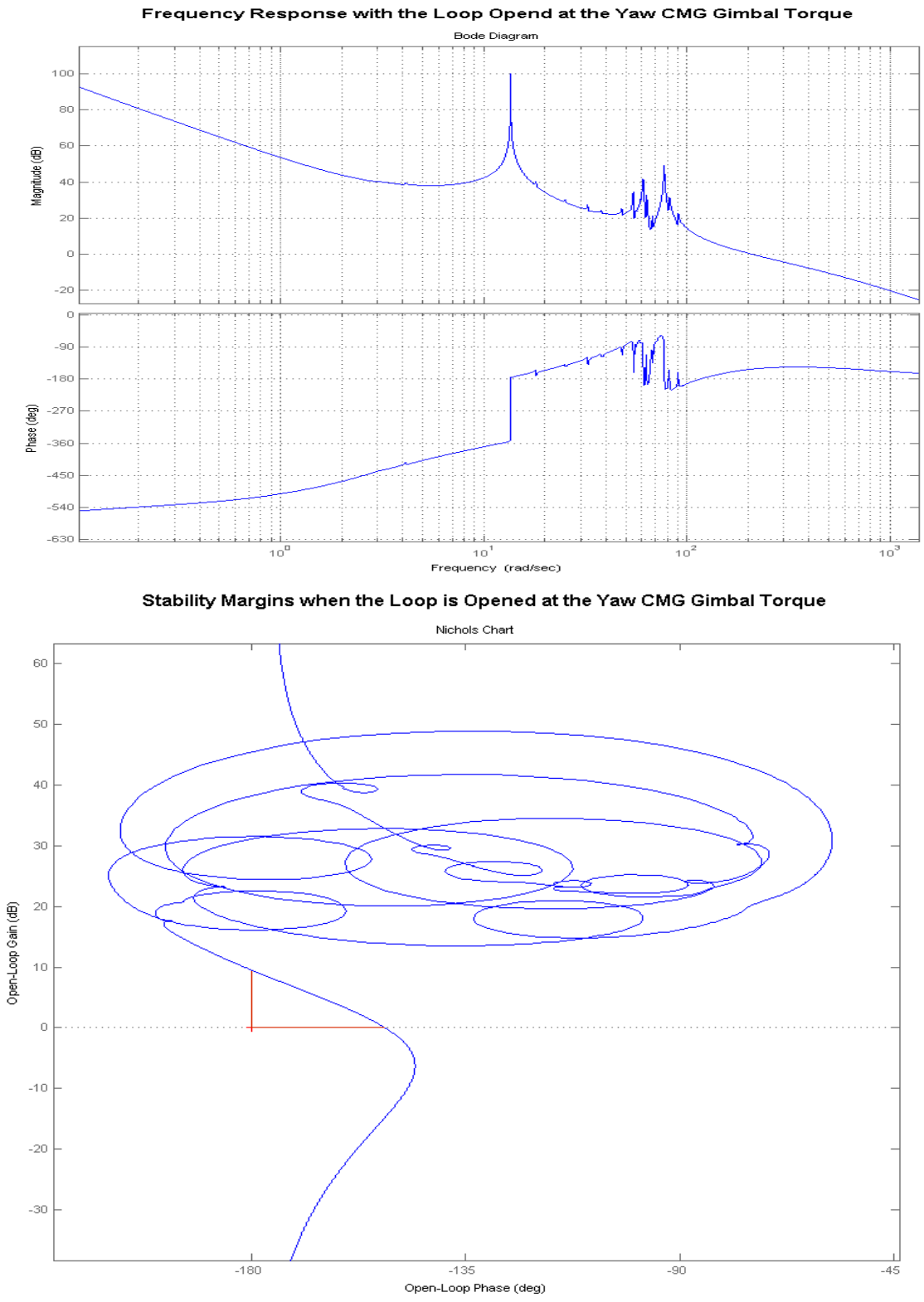


Figure 3.6.13 Frequency response with the loop opened at the yaw CMG gimbal showing stability margins

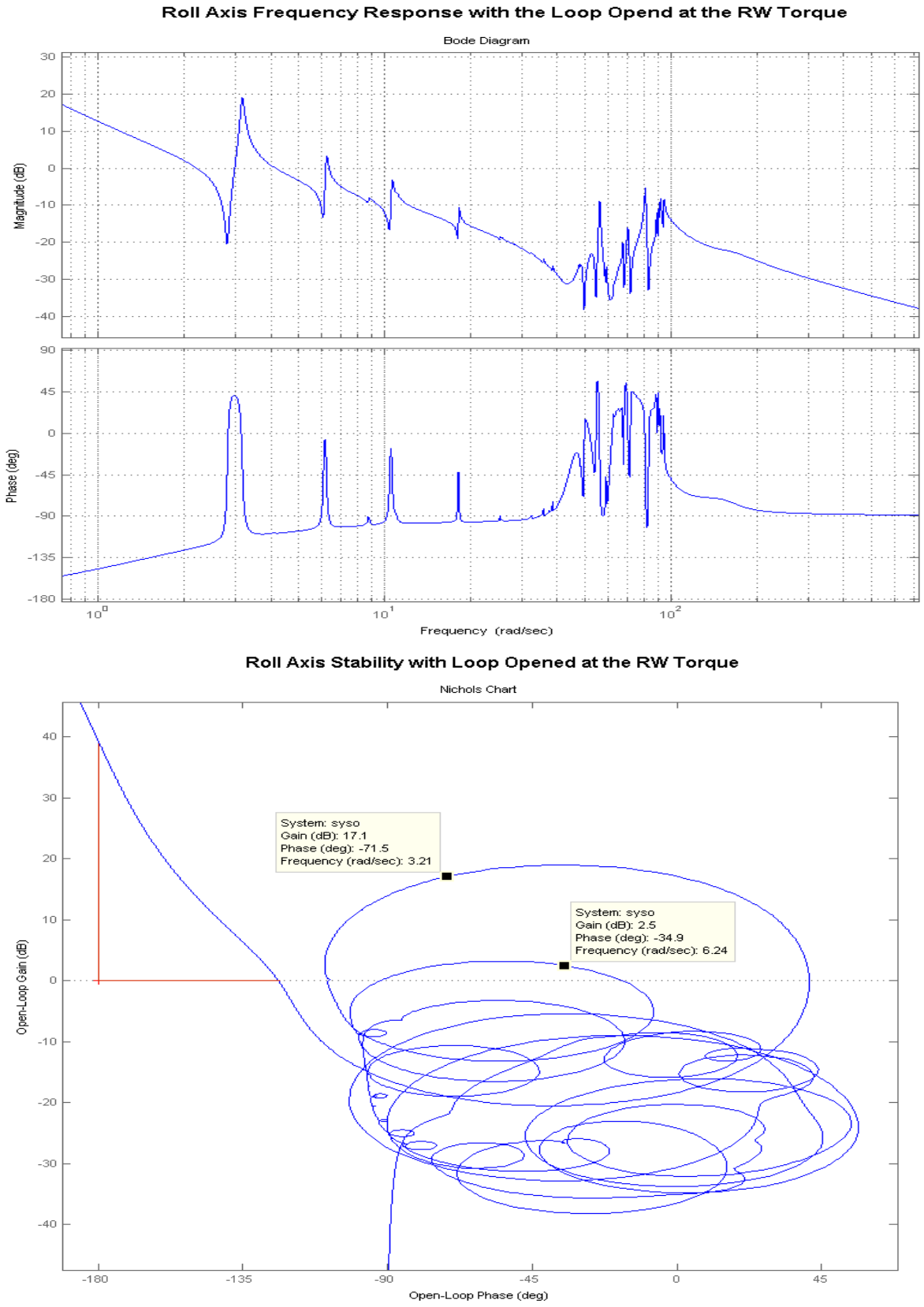


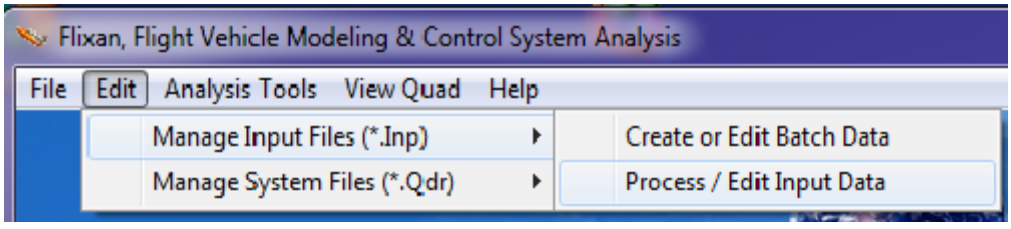
Figure 3.6.14 Frequency response with the loop opened at the reaction wheel torque showing stability margins, other loops are closed, showing that roll bandwidth is much smaller than pitch and yaw.

3.7 Using the Flight Vehicle Modeling Program to Analyze the Spacecraft with 2 SGCMG and 1 RW

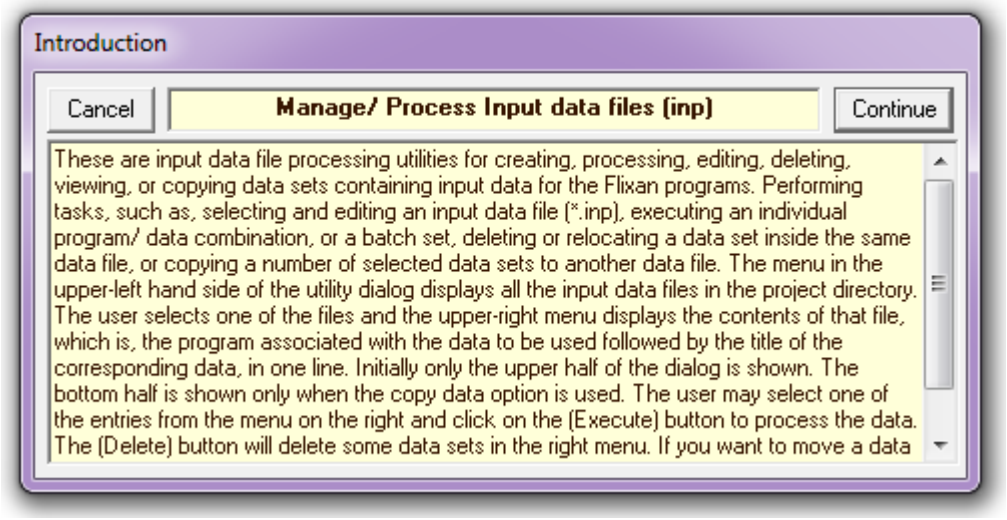
Let us now repeat the previous spacecraft analysis using the Flixan Flight Vehicle Modeling program to model the flexible spacecraft with two SG-CMGs and one Reaction Wheel. Our purpose now is to demonstrate that we can obtain the same results using state-space models derived from Flixan and to bypass the Matlab dynamic models created in folder: "*(d) Flex SC with 2SGCMG+RW ACS*". This time we will skip the detailed spacecraft data creation process and use instead the existing input data files and run them using Flixan program in batch mode to create the spacecraft state-space models. The data for this analysis is in folder: "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(f) 2SGCMG 1RW using FVP*". The input data file is "*FlxSc_2CMG_RW.Inp*" the systems file is "*FlxSc_2CMG_RW.Qdr*". The input data file contains data for the spacecraft with 2 SGCMGs for pitch and yaw control and one RW for roll control. There is a rigid body model and a flex model. It contains also a set of modal data consisting of 40 selected flex modes that were extracted from the modal data files "*FlexSc_FEM.Mod*" and "*FlexSc_FEM.Nod*" used in previous analysis. We will skip the lengthy mode selection process because it is similar to the mode selection in Section 3.5 with the addition of the reaction wheel which is located in the same node as the CMGs. The input data file also contains some data for the Flixan model reduction utility to eliminate some unused states and outputs. The Matlab analysis is performed in the sub-directory "Matan".

Creating the Systems in Batch Mode

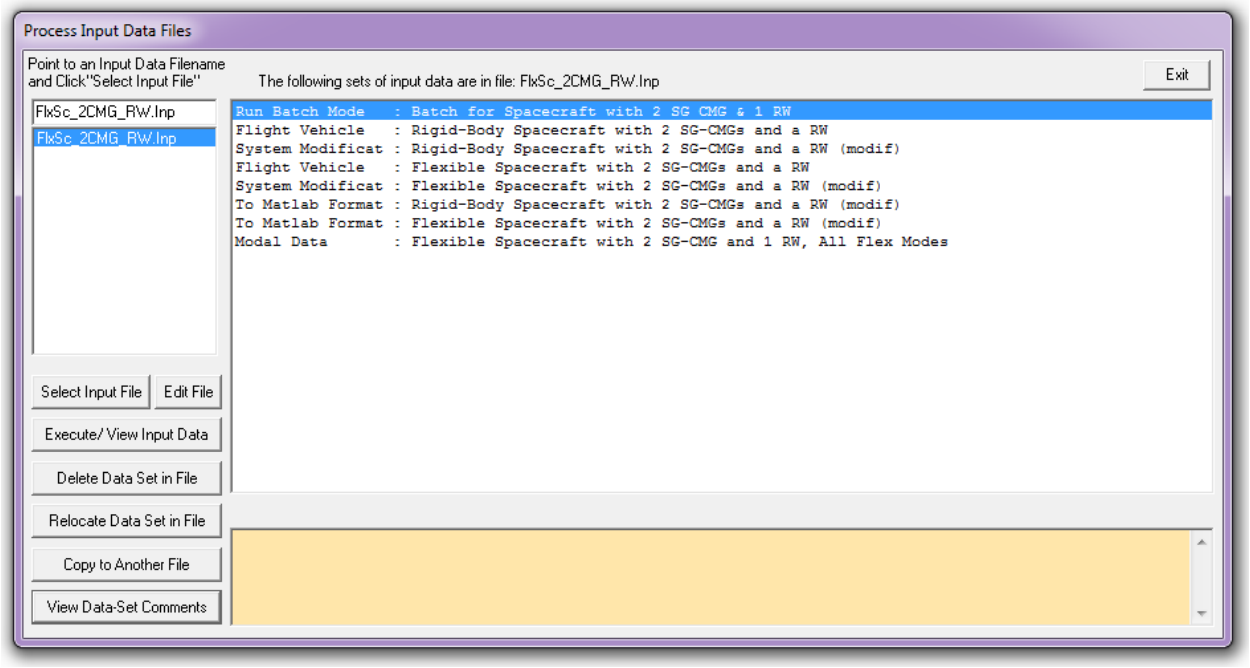
On the top of the input data file "*FlxSc_2CMG_RW.Inp*" there is a batch data-set "*Batch for Spacecraft with 2 SG CMG & 1 RW*". It is a short script of commands that speeds up the generation of the spacecraft systems. To execute the batch, you must first start the Flixan program, go to folder "*…Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(f) 2SGCMG 1RW using FVP*". Go to "*Edit*", "*Manage Input Files*", and then "*Process/ Edit Input Data*".



When the following dialog appears, click "Continue".



From the following dialog select the input file "*FlxSc_2CMG_RW.Inp*" and press the "*Select Input File*" button. The menu on the right shows all the data-sets which are saved inside the input file. Select the top batch set: "*Batch for Spacecraft with 2 SG CMG & 1 RW*", and click on "*Execute/ View Input Data*".



The Flixan program batch processor uses various utilities to process all the data-sets called by the batch and saves the spacecraft systems in file "*FlxSc_2CMG_RW.Qdr*". It creates also two state-space system m-files for the spacecraft with the two CMGs and a RW that can be loaded into Matlab: a rigid-body model in file "*sc_cmg_rw_rb.m*" and a flex spacecraft model "*sc_cmg_rw_flex.m*". Click "Exit" to end the Flixan program. The two files are transferred to the subfolder "Matan" to be loaded into Matlab for further analysis.

The Simulation Model

The simulation model used in this analysis is "*PID_RW_2SGCMG_FVP.mdl*", shown in Figure (3.7.1). It is similar to the non-linear model shown in the previous section but is missing all the non-linearities in the attitude control system and in the spacecraft dynamics. It uses Euler angles for attitude instead of quaternion. The integrators in the PID are "on". It is not intended for large angle maneuvers but for analyzing stability and performance at nominal operating conditions, such as, in the initial condition when the gimbal angles are at zero and the CMG array momentum is zero.

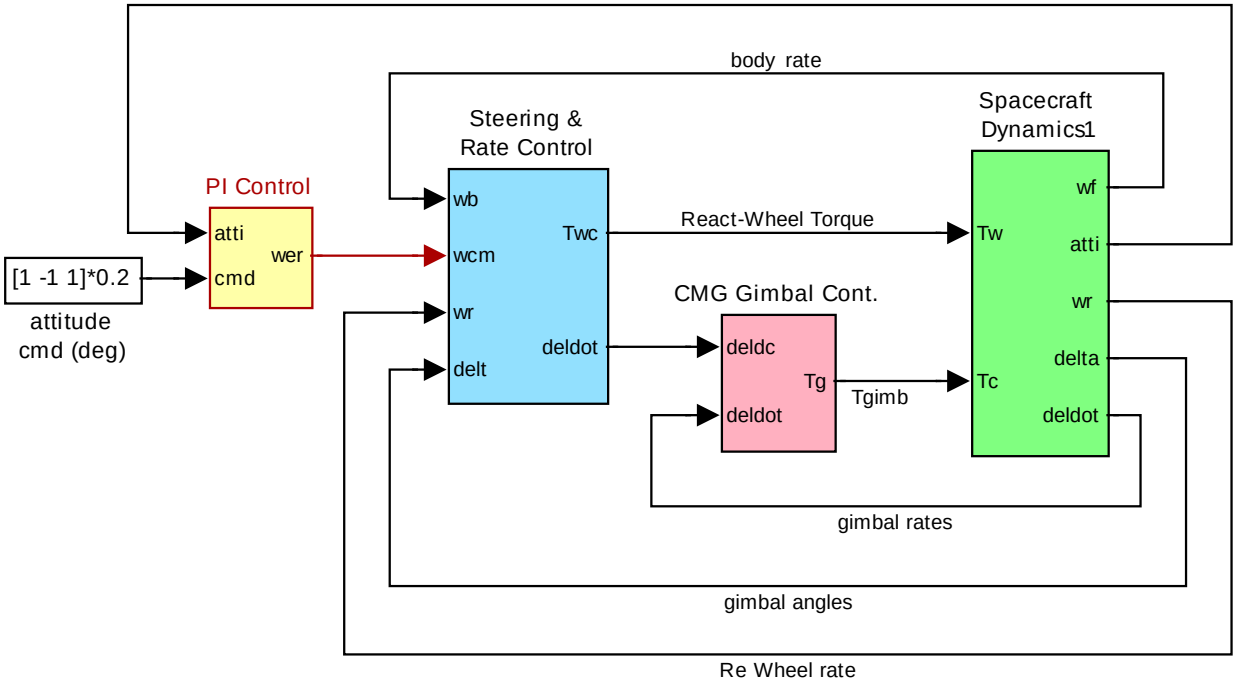


Figure 3.7.1 Linear Simulation Model in file "PID_RW_2SGCMG_FVP.mdl"

The spacecraft plus CMG dynamics (green block) contains the state-space system from file "*sc_cmg_rw_flex.m*", title: *"Flexible Spacecraft with 2 SG-CMGs and a RW (modif)"*, which was created by Flixan FVP. The system is loaded into Matlab workspace by the initialization file "start.m". It implements the linearized equations of a spacecraft with SGCMG and RW described in Section (3.2). The spacecraft block is shown in detail in Figure (3.7.2). The inputs are: pitch and yaw CMG gimbal torques, roll RW torque, and disturbance (not used). The gimbal torques come from the gimbal servo system that controls the CMG gimbal rates. The gimbal servo system and the steering were described earlier. The spacecraft outputs are: attitude, rates, and accelerations at different locations, including flexibility. There is also gimbal rates, gimbal angles, reaction wheel rate, and CMG array momentum in body axis. The initialization file "start.m" is also initializing the gimbal control system gains, the mass properties, the CMG orientation angles, gimbal directions, and momentum reference directions which are used by the steering logic.

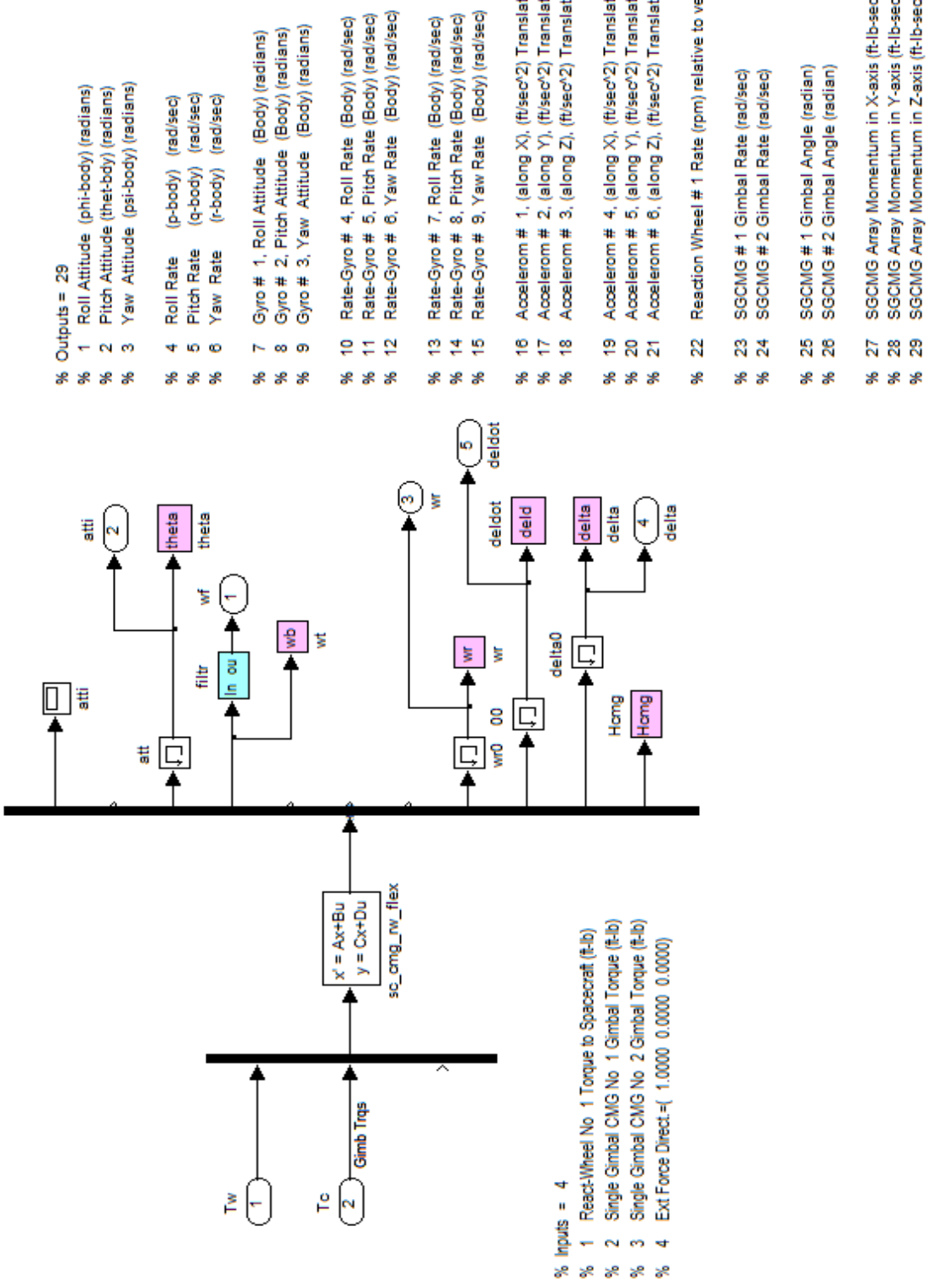


Figure 3.7.2 Spacecraft Dynamics from State-Space System: "*sc_cmg_rw_flex.m* "

The attitude controller is shown in Figure 3.7.3. It has been reduced to a PID by removing the phase-plane logic, plus rate and energy limiting non-linearities, suitable for small angles steady-state operations. This is when "kay" switches from one to zero at the completion of the phase-plane mode of operation. The rate feedback loop of the PID controller is implemented in the steering block. There is a scaling gain of 0.5 included to accommodate the previously used PI gains which were designed based on a quaternion error feedback. There is a factor of two difference between attitude error in (rad) and quaternion error.

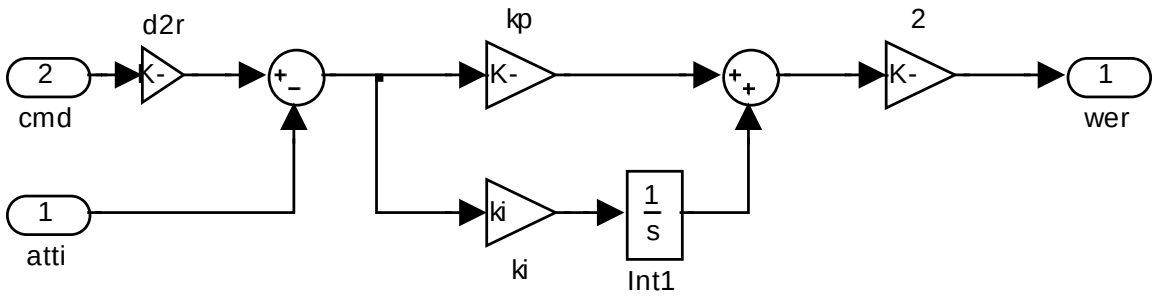


Figure 3.7.3 PI Attitude Control System

Simulation Results

The following results were obtained from the simulation model by commanding 0.2 degree attitude rotation in all 3 axes 0.2*[1 -1 1]. This model is not intended to be used for large angle maneuvers because it is missing the non-linear controls which limit rates and torques and it will generate unrealistic amounts of rates and torques. It operates entirely in PID mode and it is intended for evaluating the spacecraft stability and performance under small angles excitation. The results show a step response time of about 1.5 seconds in pitch and yaw. The roll axis is slower by design as it was already discussed. There is also more structural flexibility in roll (blue). The reaction wheel accelerates to provide the torque required to rotate the spacecraft in roll. The RW torque is limited to 10 (ft-lb). The attitude error in roll is affected by flexibility but it eventually decays due to modal damping. The CMG gimbals respond to the pitch and yaw commands and the CMG momentum is strictly in pitch and yaw. There is no CMG momentum used for roll control because roll is controlled by the reaction wheel.

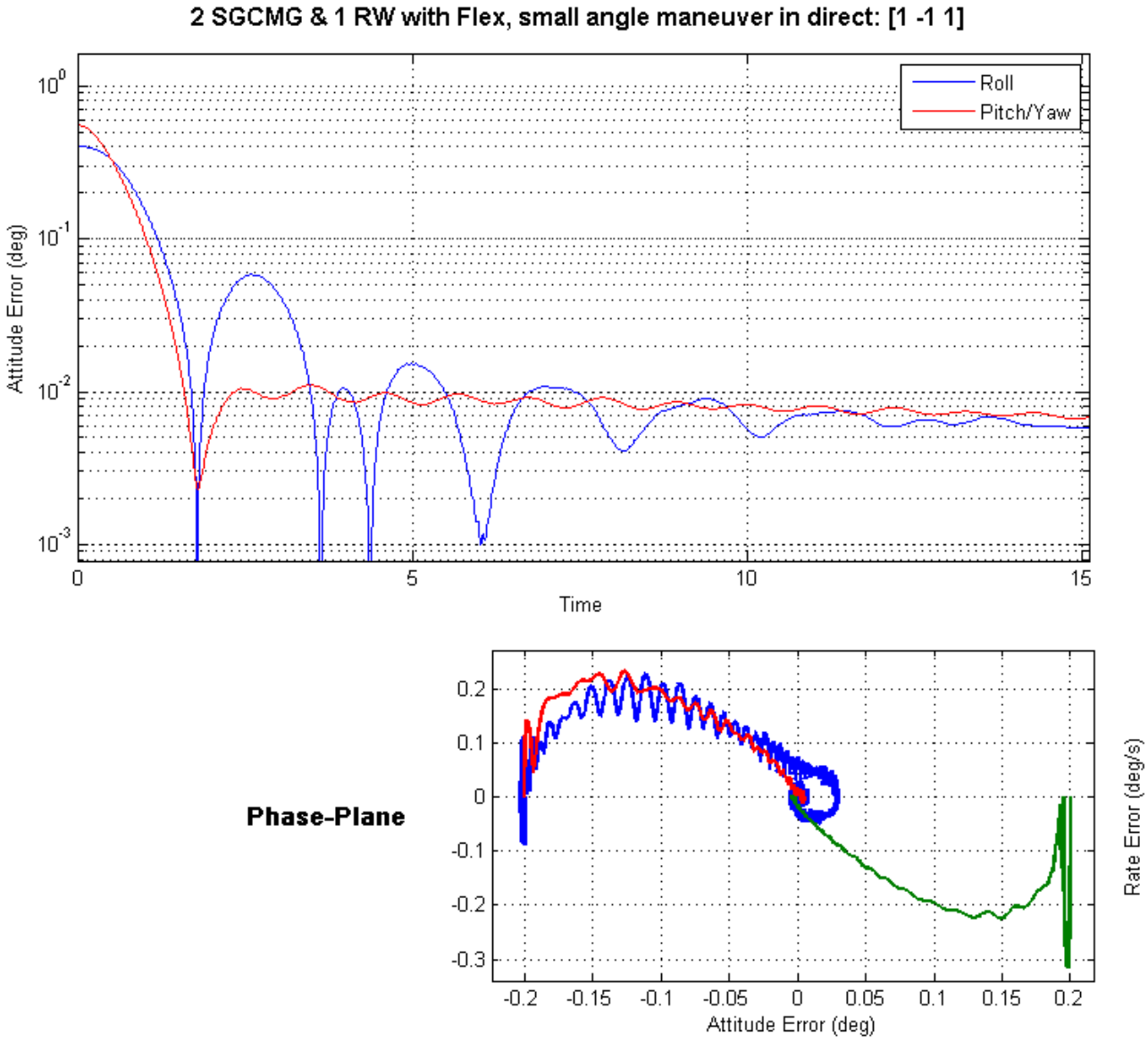


Figure 3.7.3(a) Attitude error in roll (blue) and in pitch/yaw combined (red)

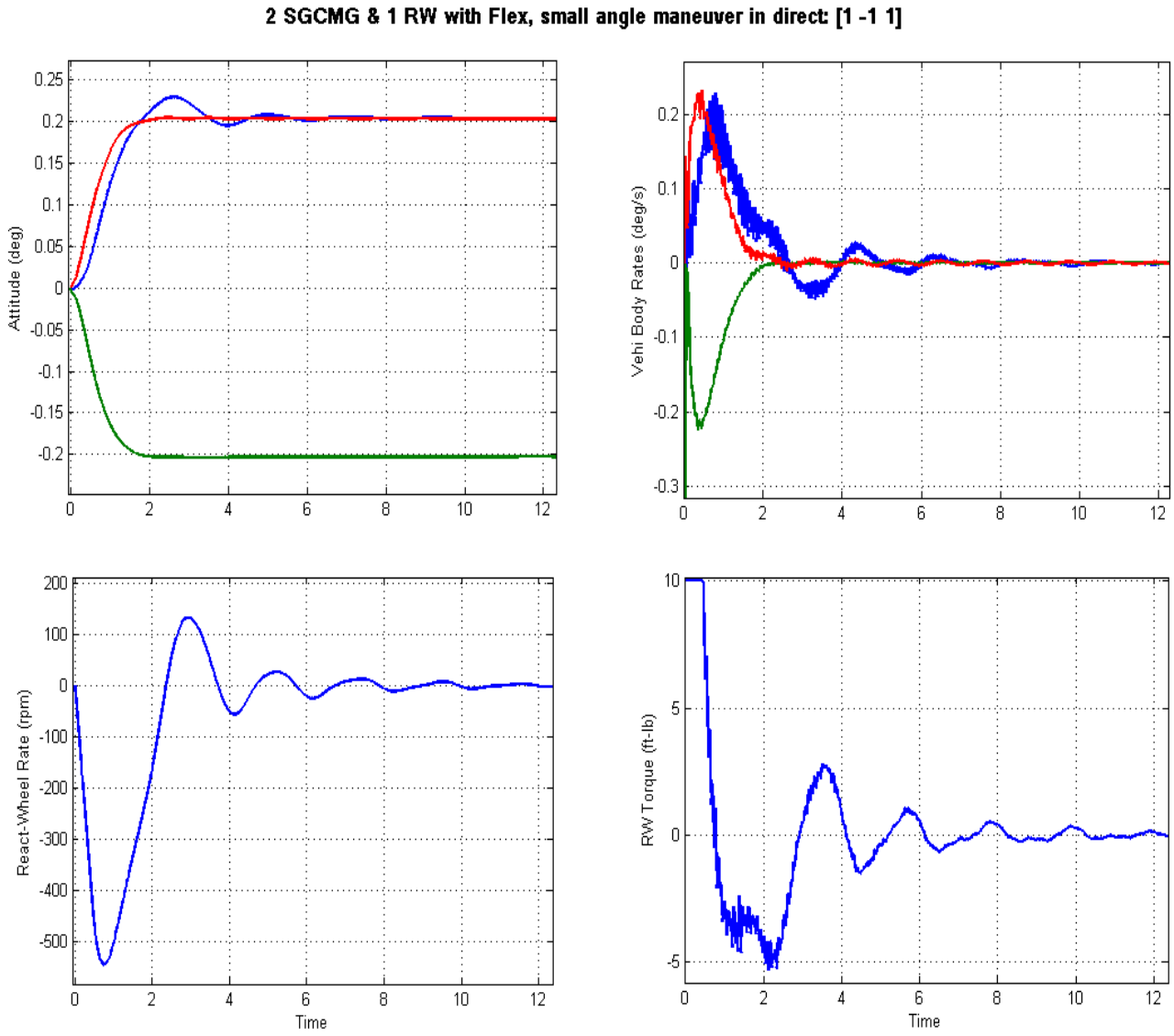


Figure 3.7.3(b) Step Response is 1.5 seconds in pitch and yaw. The roll loop is slower and it is more vulnerable to flexibility. The RW rate responds to the roll attitude command. The RW torque is limited to 10 (ft-lb)

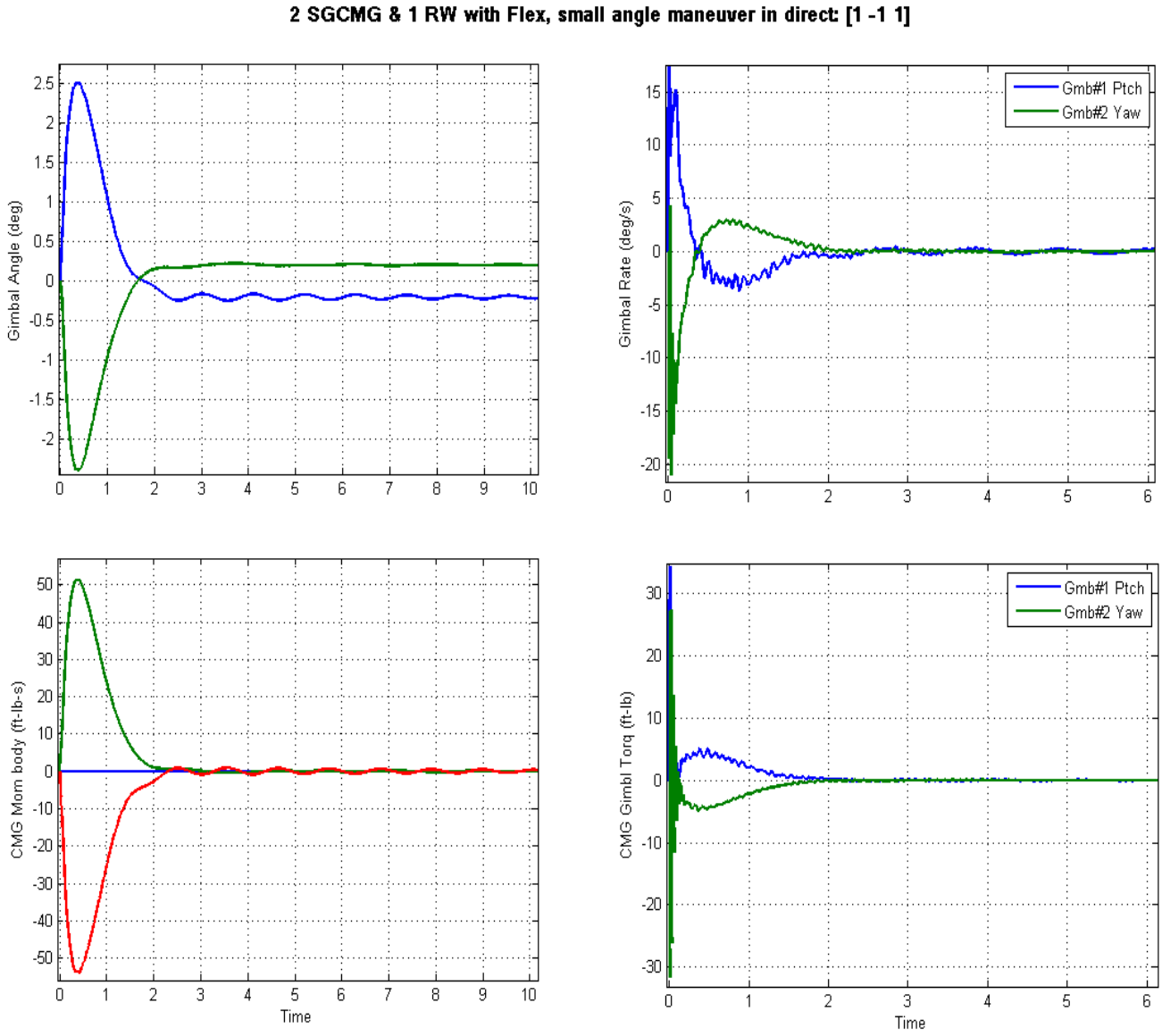


Figure 3.7.3(c) CMG Gimbal angles and rates due to the maneuver. The CMG momentum was used only in pitch and yaw, not in roll. The roll axis is controlled by the RW.

Stability Analysis

The Simulink model "*Open_Loop_FVP.mdl*" in Figure (3.7.4) is used for open-loop frequency response analysis and it consists of the same subsystems as in Figure (3.7.1). The control loops are broken for frequency response analysis at two places: (a) the reaction wheel torque output for roll, and (b) at the pitch and yaw CMG gimbal torques. Only one axis is cut at a time while the other two remain closed. In the example configuration shown below the RW torque loop is opened but the two CMG gimbal torques are closed. The Matlab file “freq.m” linearizes this model and calculates the frequency response between the input and the output and plots the Bode and the Nichols plots. The stability analysis plots are shown in Figure (3.7.5). Stability is measured by the phase and gain margins from the red cross.

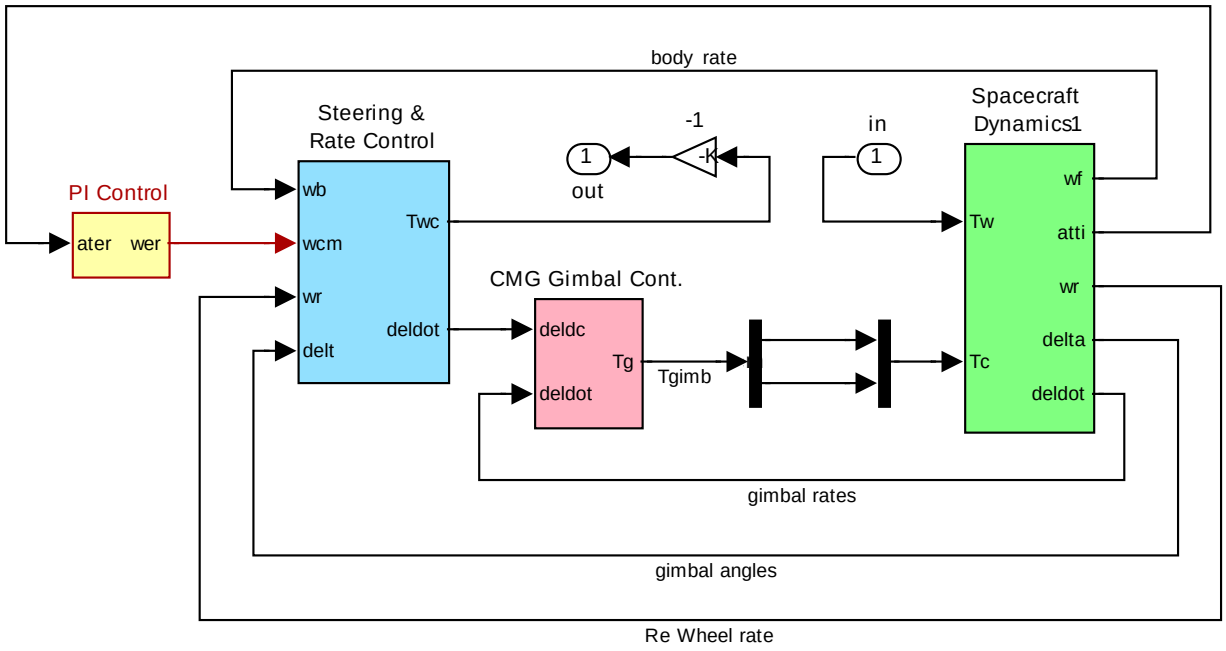
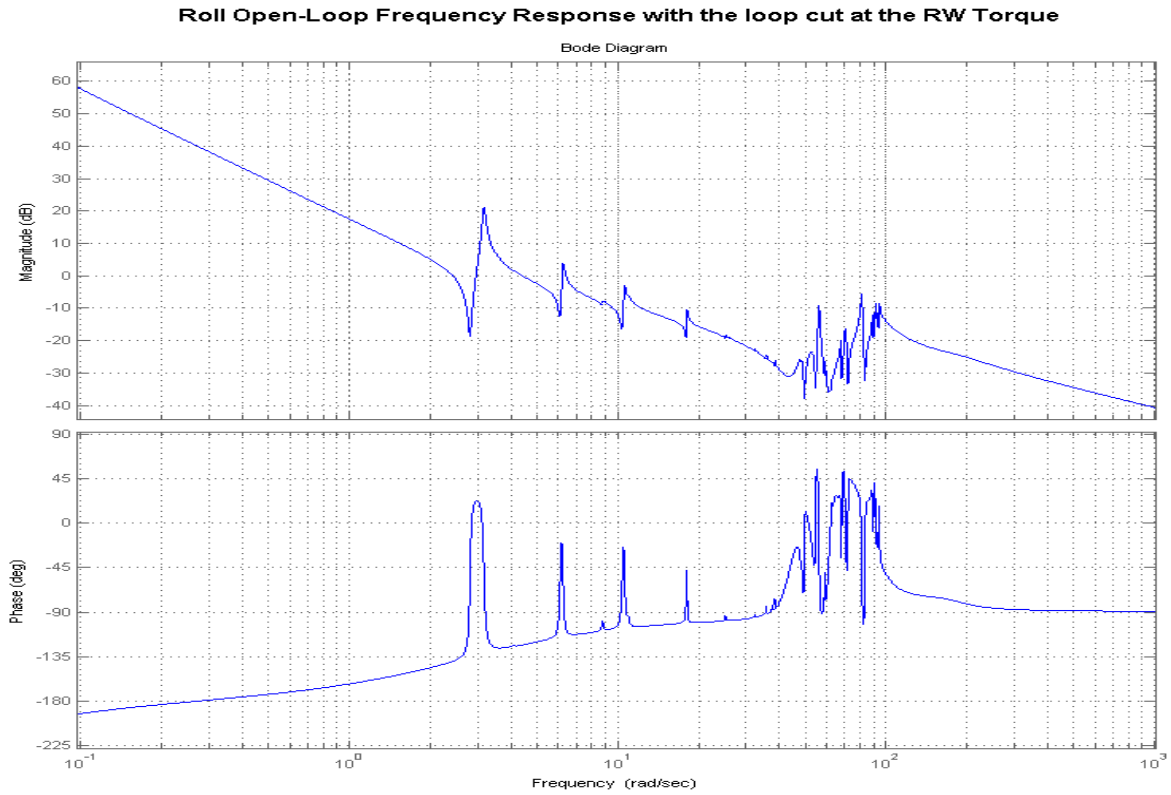


Figure 3.7.4 Simulink Model "Open_Loop_FVP.mdl" Used for Frequency Response Analysis

The frequency response and stability analysis results shown in Figures 3.7.5 are identical to the frequency response results obtained in the previous section in Figures 3.6.12, concluding, that the state-space models generated by the Flixan program are identical to the detailed dynamic models presented in the Section 3.6.



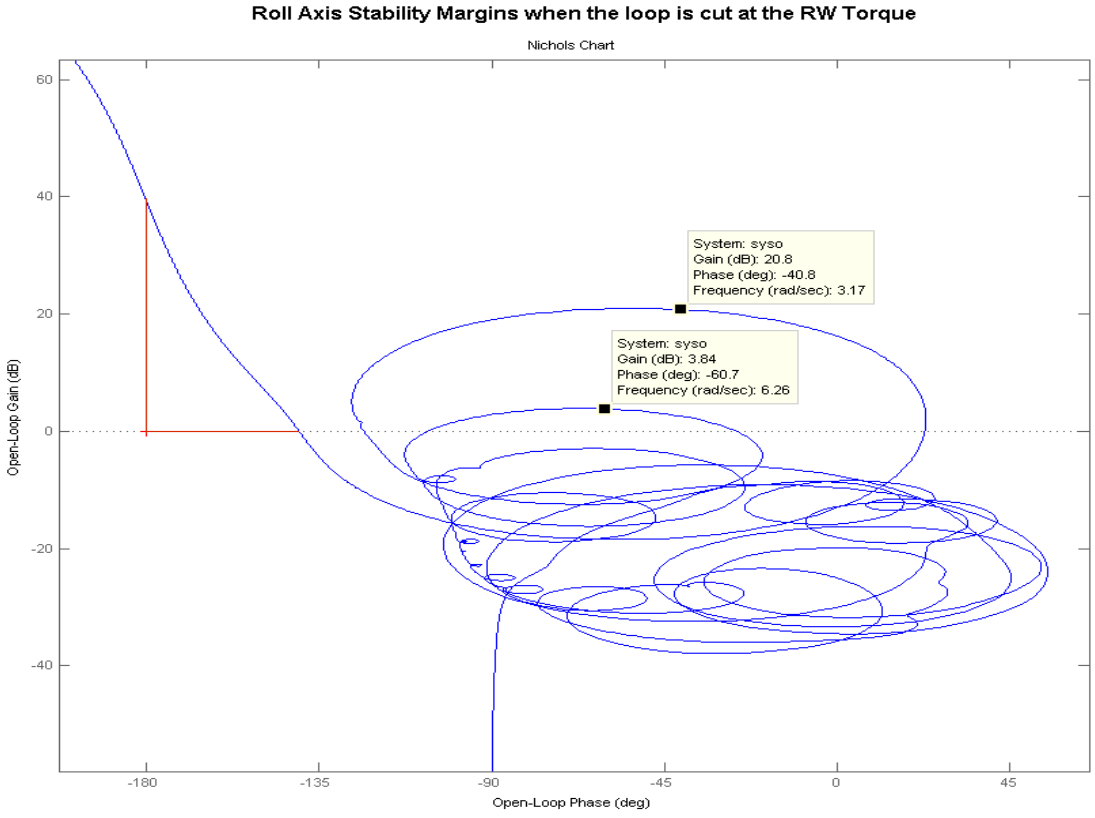
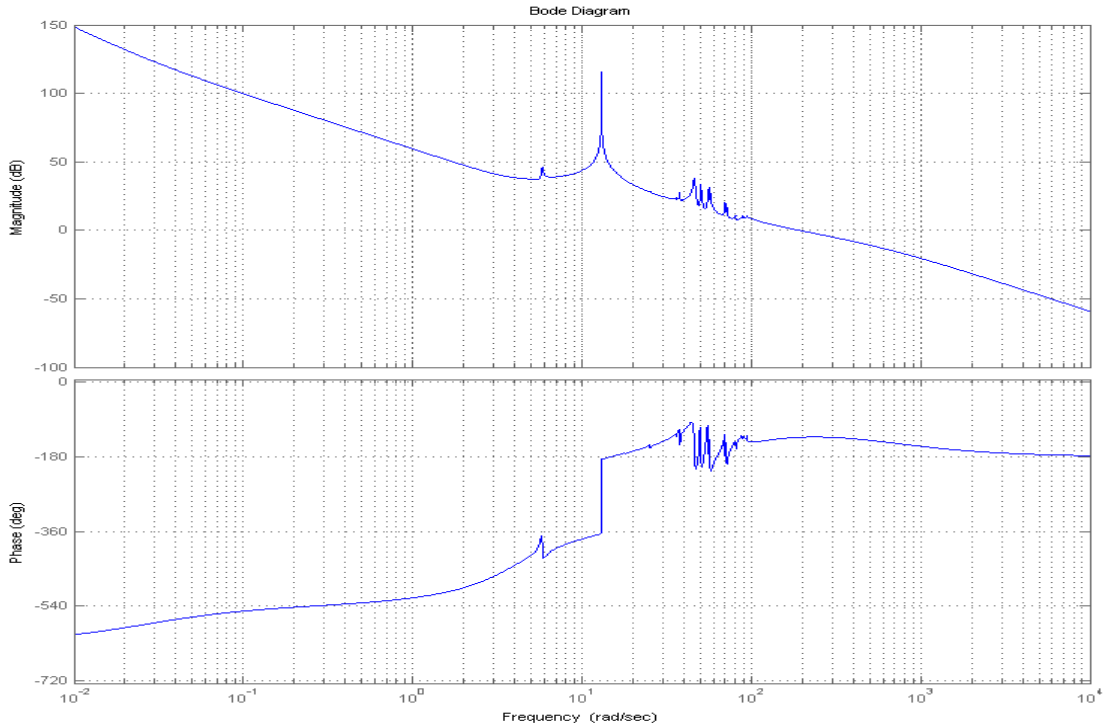
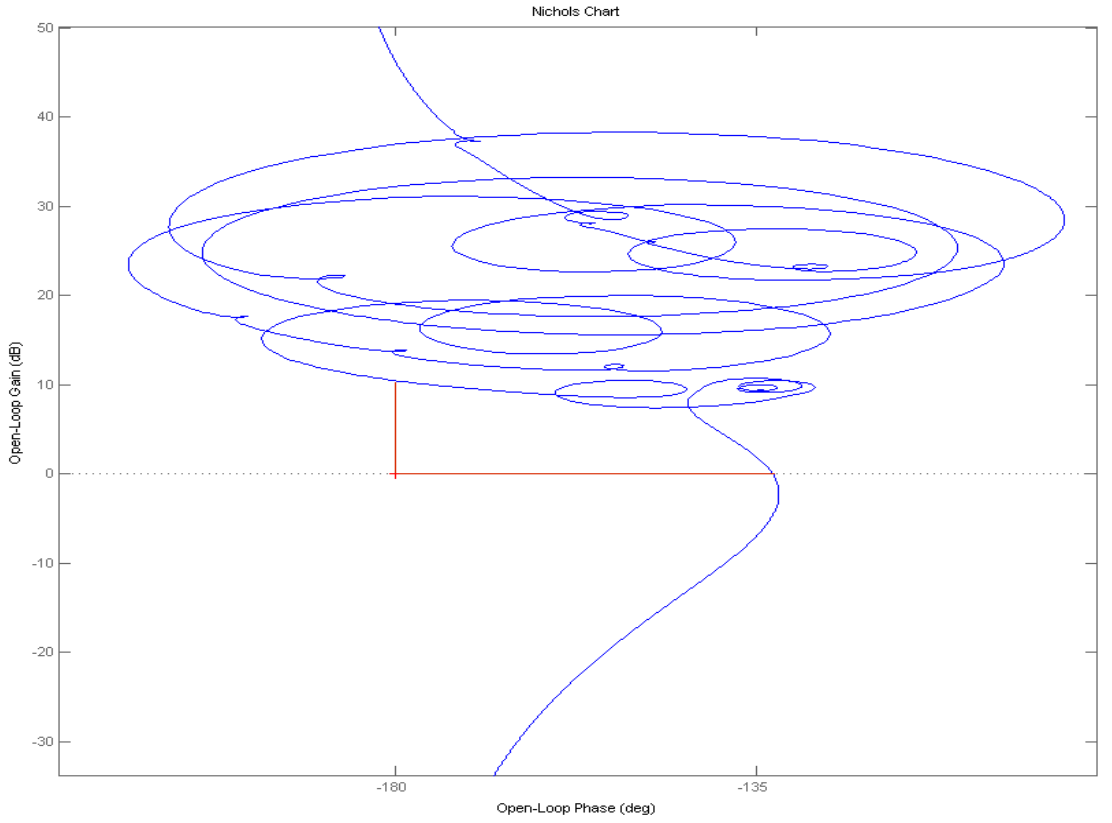


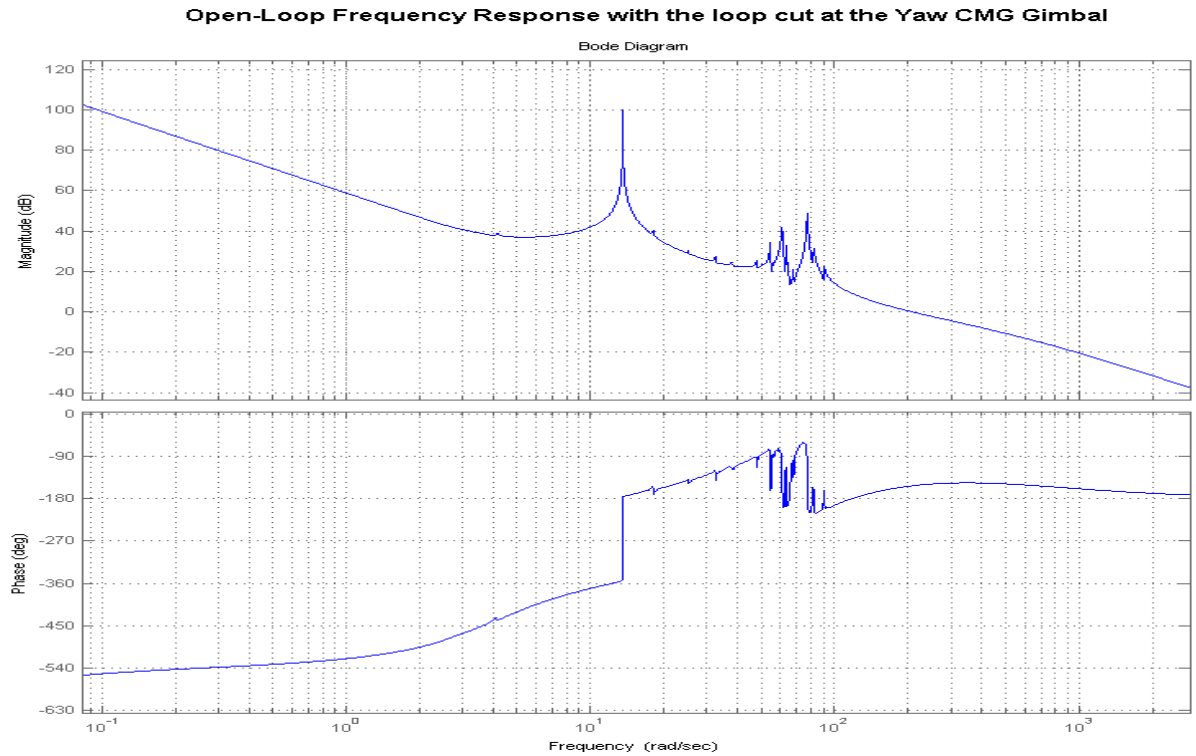
Figure 3.7.5(a) Open-Loop Frequency Response and Stability Analysis with loop opened at the RW Torque

Open-Loop Frequency Response with the loop cut at the Pitch CMG Gimbal



Stability Margins when the loop is cut at the Pitch Gimbal





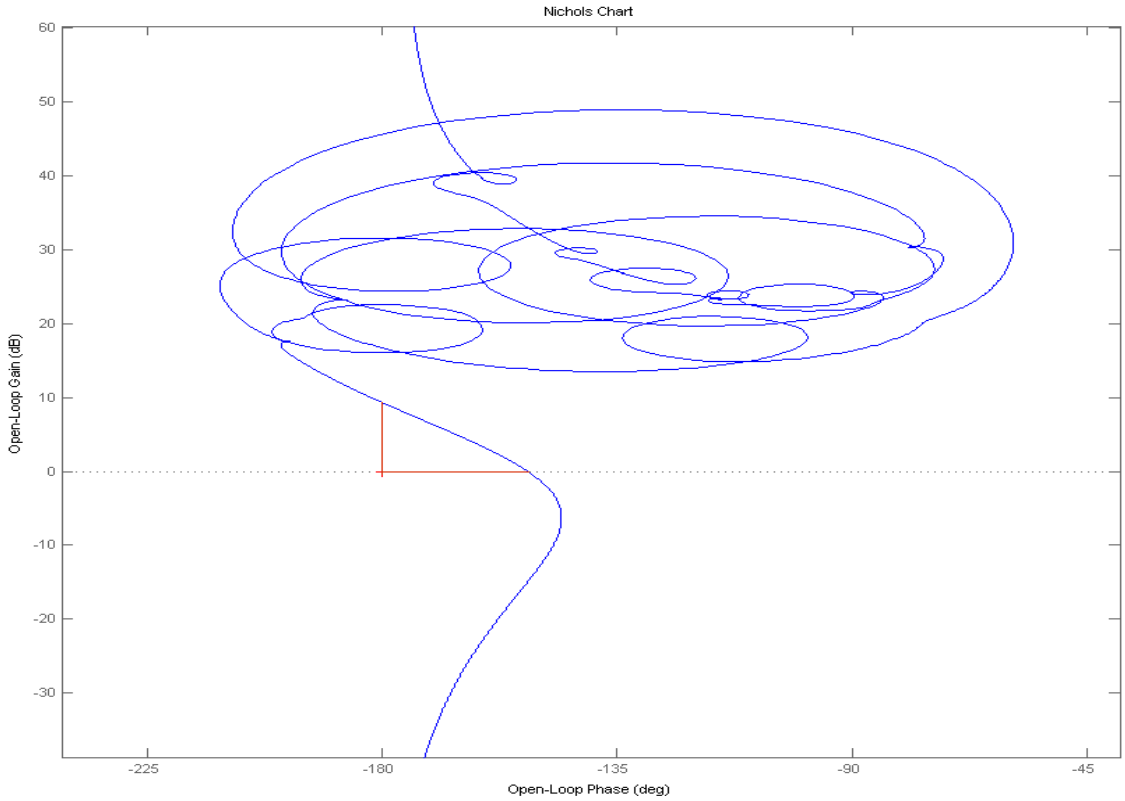


Figure 3.7.5(c) Open-Loop Frequency Response and Stability Analysis with loop opened at the Yaw CMG Gimbal Torque